ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

О.В.Шишацька
А.В.Криволап
А.В.Шишацький

# ТЕОРІЯ ПРОГРАМУВАННЯ
# ПРАКТИКУМ

Навчальний посібник

Київ 2024

УДК 519.8 (075.8)
ББК 22.18я73
Н62

Рецензенти
д.ф.-м.н., професор Нікітченко М.С. (професор кафедри теорії та технології програмування Київського національного університету імені Тараса Шевченка)
к.ф.-м.н., доцент Шкільняк О.С. (доцент кафедри факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка)

Рекомендовано до друку вченою радою факультету комп'ютерних наук та кібернетики (протокол № ?? від "??" ?????? 2024 року)

Ухвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики (протокол № ??? від ???? 2024 року )

к.ф.-м.н., доц. Шишацька Олена Володимирівна
к.ф.-м.н. ас. Криволап Андрій Володимирович
ас. Шишацький Андрій Вікторович
Теорія програмування. Практикум: навчальний посібник / Електронне видання, - 2024. – ?? с.

Викладено матеріали до практичних занять з обов'язкової навчальної дисципліни «Теорія програмування». В посібнику наведено умови практичних робіт та теоретичний матеріал для їх виконання, завдання для самостійної роботи та практичних занять, перелік контрольних запитань та базових задач до кожної теми.

Для студентів третього курсу факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка, які навчаються за освітньо-професійною програмою «Інформатика» спеціальності 122 «Комп'ютерні науки».

**Зміст**

## ВСТУП

Навчальна дисципліна **«Теорія програмування»** є складовою освітньо-професійної програми підготовки спеціалістів за освітньо-кваліфікаційним рівнем «бакалавр» галузі знань 12 „Інформаційні технології" зі *спеціальності* 122 „Комп'ютерні науки", *освітньо-професійної програми* – „Інформатика".

Дана дисципліна є обов'язковою навчальною дисципліною за програмою "Інформатика".

Викладається в 5 семестрі 3 курсу бакалаврату в обсязі 120 годин.

(*4 кредити ECTS*) зокрема: *лекції – 28 год., консультації – 2 год., практичних занять – 28 год., самостійна робота – 62 год.* У курсі передбачено 2 *частини* та 2 *контрольні роботи.* Завершується дисципліна – іспитом в 5 семестрі.

**Мета дисципліни** – засвоєння основних теоретичних концепцій, принципів та понять сучасного, зокрема композиційного, програмування; методів формалізації мов програмування та доведення коректності програм.

**Попередні вимоги до опанування або вибору навчальної дисципліни (*за наявності*):**

1. *Знати*: основні поняття, засоби і методи програмування, їх застосування в інформатиці; знати мови програмування та логіки 1-го порядку, їх можливості для опису предметних областей.
2. *Вміти*: описувати на формальних мовах 1-го порядку твердження стосовно тих чи інших предметних областей та властивостей програм; встановлювати істинність пропозиційних формул, формул 1-го порядку.
3. *Володіти елементарними навичками*: програмування в сучасних мовах, тестування програм, перевірки виконуваності формул.

**Завдання (навчальні цілі):** набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень у програмуванні, відповідно освітньої кваліфікації «Бакалавр з комп'ютерних наук». Зокрема розвивати:

- здатність до абстрактного мислення, аналізу та синтезу;
- здатність до математичного формулювання та досліджування неперервних та дискретних математичних моделей, обґрунтовування вибору методів і підходів для розв'язування теоретичних і прикладних задач у галузі комп'ютерних наук, аналізу та інтерпретування;
- здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно- орієнтованого, функціонального, логічного, з відповідними моделями, метода-

ми й алгоритмами обчислень, структурами даних і механізмами управління.

Для допуску до дисципліни **„Теорія програмування”** освітньо-професійної програми «Інформатика» студент повинен опанувати компетентності та результати навчання, які надають дисципліни «Дискретна математика», «Математична логіка» та «Теорія алгоритмів» програми «Інформатика».

В результаті вивчення навчальної дисципліни студент повинен:
**знати**: основні поняття теорії програмування, методи формалізації мов програмування, зокрема формалізації та аналізу семантики та синтаксису програм; теорію найменших нерухомих точок;

**вміти**: формалізувати синтаксис мов програмування за допомогою БНФ та граматик, робити синтаксичний аналіз програм, будувати семантичний терм програми в алгебрі програм, доводити коректність програм. Формалізувати та досліджувати рекурсивні програми.

| **Результат навчання (1.знати; 2.вміти; 3.комунікація; 4.автономність та відповідальність)** | | **Форми викладання і навчання** | **Методи оцінювання та порогові критерії оцінювання (за необхідності)** | **Відсоток у загальній оцінці з дисципліни** |
|---|---|---|---|---|
| Код | Результат | | | |
| РН1.1 | Знати основні поняття програмування. | Лекції, практичні заняття | Контрольна робота 60% правильних відповідей, проєктне завдання, домашні завдання, іспит | 15% |
| РН1.2 | Знати методи формалізації мов програмування. | Лекції, практичні заняття | Контрольна робота 60% правильних відповідей, проєктне завдання, домашні завдання, іспит | 15% |
| РН1.3 | Знати методи моделювання предметних областей. | Лекції, практичні заняття | Контрольна робота 60% правильних відповідей, проєктне завдання, домашні завдання, | 40% |
| РН2.1 | Вміти формалізувати мови програмування, моделювати предметні області за допомогою відповідних мов, засвоювати засоби аналізу програм. | Лекція, практичні заняття, самостійна робота | Поточне оцінювання, проєктне завдання, домашні завдання, іспит | 10% |
| РН3.1 | Обгрунтовувати власний погляд на задачу, спілкуватися з колегами з питань проєктування, розробки специфікацій та програм. | Лекції, практичні заняття | Поточне оцінювання, проєктні завдання, іспит | 10% |
| РН4.1 | Організовувати свою самостійну роботу для досягнення результату. | Практичні заняття, самостійна робота | Поточне оцінювання, проєктне завдання, домашні завдання, іспит | 10% |

## Співвідношення результатів навчання дисципліни із програмними результатами навчання

| **Результати навчання дисципліни**<br>**Програмні результати навчання** | **РН 1.1** | **РН 1.2** | **РН 1.3** | **РН 2.1** | **РН 3.1** | **РН 4.1** |
|---|---|---|---|---|---|---|
| (З опції освітньої програми) | | | | | | |
| ПРН1. Застосовувати грунтовні знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів впливу, аналізу, обробки та синтезу інформації в предметній області комп'ютерних наук. | + | + | + | + | + | + |
| ПРН5. Проектувати, розробляти та аналізувати алгоритми розв'язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислювальних функцій. | + | + | + | | | |

## Схема формування оцінки.
## Форми оцінювання студентів:

- семестрове оцінювання:
  1. *Контрольна робота 1: РН 1.1, РН 1.2 — 10 балів/6 балів.*
  2. *Контрольна робота 2: РН 1.1, РН1.3 - 15 балів/9 балів.*
  3. *Проєктне завдання: РН 1.1, РН 1.2, РН 2.1, РН1.3, РН 3.1 - 10 балів/6 балів.*
  4. *Домашні завдання: РН 1.1, РН 1.2, РН 2.1, РН1.3, РН 4.1 - 15 балів/9 балів.*
  5. *Поточне оцінювання: РН 2.1., РН 3.1, РН 4.1 – 10 балів/6 балів.*
- **підсумкове оцінювання (у формі іспиту)**:
  - *максимальна кількість балів які можуть бути отримані студентом: 40 балів/ 24 бали;*
  - *результати навчання які будуть оцінюватись: РН1.1, РН1.2, РН1.3, РН2.1, РН3.1, РН4.1;*
  - *форма проведення і види завдань: письмова робота.*

***Види завдань:*** 4 письмових питання.

- 1 питання (теоретичне): *РН1.1, РН3.1, РН4.1*;
- 2 питання (теоретичне): *РН1.2, РН3.1, РН4.1*;
- 3 питання (практичне): *РН1.3, РН3.1, РН4.1*;
- 4 питання (практичне): *РН2.1, РН3.1, РН4.1*;

**За розгорнуту відповідь на кожне завдання студент може отримати від 1 до 10 балів.**

**Критерії оцінювання відповіді студента на питання:**

повнота розкриття питання 1-4 бали;

логіка викладення 2 бал;

аналітичні міркування 1-4 бали.

**Організація оцінювання**

**Терміни проведення форм оцінювання:**

1. *Контрольна робота 1: до 5 тижня семестру.*
2. *Контрольна робота 2: до 12 тижня семестру.*
3. *Проєктна робота: до 12 тижня семестру.*
4. *Домашні завдання: протягом семестру.*
5. *Поточне оцінювання: протягом семестру.*

Студент має право на одне перескладання кожної контрольної роботи із можливістю отримання максимально 80% початково визначених за цю контрольну роботу балів. Термін перескладання визначається викладачем.

Проєктне завдання студент має право замінити на здачу сертифікату з можливістю отримання 9 балів. Для реалізації цього права студент повинен до 1 жовтня поточного навчального року написати заяву (classroom), в якій надати інформацію про обраний курс. Курс має бути попередньо узгоджений з викладачем. Дата отримання сертифікату повинна належати часовому проміжку від 01 вересня до 01 грудня поточного навчального року.

| **Шкала відповідності оцінок** | |
|---|---|
| **Відмінно** / Excellent | 90-100 |
| **Добре** / Good | 75-89 |
| **Задовільно** / Satisfactory | 60-74 |
| **Незадовільно** / Fail | 0-59 |

У посібнику викладено матеріали до практичних занять з обов'язкової навчальної дисципліни «Теорія програмування». В посібнику наведено умови практичних робіт та теоретичний матеріал для їх виконання, завдання для самостійної роботи та практичних занять, перелік контрольних запитань та базових задач до кожної теми.

# 1 СИНТАКСИС ТА СЕМАНТИКА ПРОГРАМ

## 1.1 ПРАКТИЧНА РОБОТА 1. Синтаксис програм. Доведення синтаксичної коректності програм

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Формалізація простої мови програмування. Неформальний опис простої мови програмування. Формальний опис синтаксису мови SIPL (сторінка 6-12).

Використовуватимемо просту мову програмування SIPL від характеристики SImple Programming Language (Проста Мова Програмування). Говорячи неформально, мова SIPL має числа та змінні цілого типу, над якими будуються арифметичні вирази та умови. Основними операторами є: присвоювання, послідовне виконання, розгалуження, цикл.

Мова SIPL може розглядатися як надзвичайно спрощена традиційна мова програмування. У мові SIPL відсутні складні типи даних, оператори введення-виведення, процедури та багато інших конструкцій традиційних мов програмування. Також немає явної типізації. Разом з тим, ця мова досить потужна для програмування різних арифметичних функцій, більше того, у ній можуть бути запрограмовані всі обчислювані функції над цілими числами.

Синтаксис мови SIPL можна задати за допомогою такої БНФ:

| **Ліва частина правила – метазмінна** | **Права частина правила** | **Ім'я правила** |
|---|---|---|
| <програма>::= | begin <оператор>end | NP1 |
| <оператор>::= | <змінна>:= <вираз>\|<br><оператор>; <оператор>\|<br>if <умова>then <оператор><br>else <оператор>\|<br>while <умова>do <оператор>\|<br>begin <оператор>end \|<br>skip | NS1<br>NS2<br>NS3<br>NS4<br>NS5<br>NS6 |
| <вираз>::= | <число>\| <змінна>\| <вираз>+ <вираз>\|<br><вираз>− <вираз>\| <вираз>* <вираз>\|<br>(<вираз>) | NA1<br>...<br>NA6 |
| <умова>::= | <вираз>= <вираз>\| <вираз>> <вираз>\|<br><умова>∨ <умова>\| ¬ <умова>\|<br>(<умова>) | NB1<br>...<br>NB5 |
| <змінна>::= | ... M \| N \| ... | NV... |
| <число>::= | ... –1 \| 0 \| 1 \| 2 \| 3 \| ... | NN... |

Наведена БНФ задає мову SIPL як набір речень (слів), що виводяться з метазмінної <програма>. Виведення можна подати у вигляді дерева.

Наведені вище позначення метазмінних не зовсім зручні, тому часто використовують форми, ближчі до математики. Уведемо позначення для всіх синтаксичних категорій і нові метазмінні, що вказують на представників цих категорій.

| Метазмінна | Синтаксична категорія | Нова метазмінна |
|---|---|---|
| <програма> | Prog | P |
| <оператор> | Stm | S |
| <вираз> | Aexp | a |
| <умова> | Bexp | b |
| <змінна> | Var | x |
| <число> | Num | n |

У наведених позначеннях БНФ мови SIPL набуває наступного вигляду

| **Ліва частина правила** | **Права частина правила** | **Ім'я правила** |
|---|---|---|
| P::= | begin S end | P1 |
| S::= | x := a \| S1 ; S2 \| if b then S1 else S2 \| while b do S \| begin S end \| skip | S1–S6 |
| a::= | n \| x \| a1 + a2 \| a1 - a2 \| a1 * a2 \| (a) | A1–A6 |
| b::= | a1 = a2 \| a1 > a2 \| b1 $\vee$ b2 \| $\neg$ b \| (b) | B1–B5 |
| x::= | . . . M \| N \| . . . \| . . . | NV. . . |
| n::= | . . . -1 \| 0 \| 1 \| 2 \| 3 \| . . . | NN. . . |

Надалі будемо користуватися введеними позначеннями для запису структури програм та їхніх складових.

### Практичне завдання (приклад)

Обчислити $x^y$, використовуючи функції $+, -, \times$ $(x, y > 0)$.
1.Написати SIPL-програму для розв'язку сформульованої задачі.
2. Побудувати дерево її синтаксичного виводу.
3. Перевірити синтаксичну правильність програми.

Напишемо SIPL-програму для обчислення фунції $x^y$, використовуючи функції $+, -, \times$ $(x, y > 0)$. Побудуємо дерево її синтаксичного виводу та перевіримо синтаксичну правильність програми.

Результат обчислення будемо зберігати у змінній $R$, яку на початку програми ініціалізуємо 1. Щоб обчислити степінь, ми маємо $y$ разів домножити дану змінну на $x$. Для цього використаємо оператор циклу. Як лічильник може бути використана власне змінна $y$, що буде зменшуватися на 1 кожну ітерацію.

Підсумовуючи, отримаємо наступну програму (варто зазначити, що обидва

оператори присвоєння в циклі, групуємо за допомогою ключових слів **begin** та **end**, так як оператор циклу має вищий пріорітет ніж оператор послідовного виконання):

```
begin
    R := 1;
    while Y > 0 do
    begin
        R := R * X;
        Y := Y - 1
    end
end
```

Побудуємо дерево синтаксичного виводу отриманої програми:

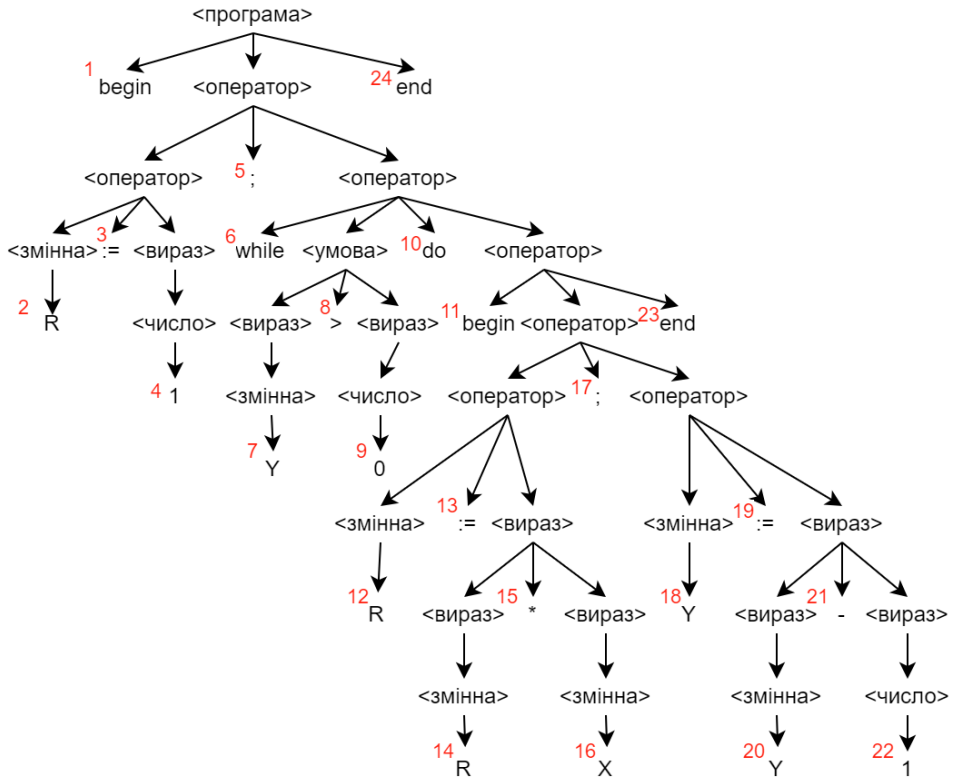


Рис. 1: Дерево синтаксичного виводу

Для перевірки синтаксичної коректності, потрібно здійснити обхід дерева синтаксичного виводу зліва направо. Отримана таким чином з термінальних символів в листках програма повинна співпадати з програмою для якої було побудовано дерево. Для цього в побудованому дереві на рис. 1 всі листки було відмічено числами, що відповідають порядку обходу. Випишемо отриману програму:

1 begin 2 R 3 := 4 1 5 ; 6 while 7 Y 8 > 9 0 10 do 11 begin 12 R 13 := 14 R 15 * 16 X 17 ; 18 Y 19 := 20 Y 21 - 22 1 23 end 24 end

Легко перевірити, що тексти програм співпадають, а отже наведена про-

грама є синтаксично коректною.

### Перелік задач

### Варіанти

1. Обчислити $x - y$, використовуючи функцію $-1$ $(x, y > 0)$.
2. Обчислити $\lfloor \lg n \rfloor$, використовуючи функції div, mod, +, – $(n > 0)$.
3. Обчислити $3^x$, використовуючи функції $\times$, +, – $(x > 0)$.
4. Обчислити $\lfloor \log_2 n \rfloor$, використовуючи функції div, mod, +, – $(n > 0)$.
5. Обчислити $(2n)!!$ $(n > 0)$.
6. Обчислити $(2n + 1)!!$ $(n > 0)$.
7. Обчислити суму арифметичної прогресії $2, 4, \ldots, 2n$, використовуючи функції +, – $(n > 0)$.
8. Обчислити суму геометричної прогресії $1, 2, 4, \ldots, 2^n$, використовуючи функції $\times$, +, – $(n > 0)$.
9. Обчислити $\lfloor \log_x y \rfloor$, використовуючи функції div, mod, +, – $(x > 1, y > 0)$.
10. Обчислити $C_x^y (x, y > 0)$.
11. Обчислити $F_n$, $n$-того елементу ряду Фібоначчі, $(n > 0)$.
12. Перевірити парність числа $n$, за допомогою циклу, не використовуючи функції div, mod $(n > 0)$.
13. Перевірити непарність числа $n$, за допомогою циклу, не використовуючи функції div, mod $(n > 0)$.
14. Перевірити чи ділиться $x$ на $y$, використовуючи функції +, – $(x, y > 0)$.
15. Перевірити простоту числа $n$, використовуючи функції div, mod, +, – $(n > 0)$.
16. Обчислити $\lfloor \sqrt{n} \rfloor$, використовуючи функції $\times$, +, – $(n > 0)$.
17. Обчислити $5^x$, використовуючи функції $\times$, +, – $(x > 0)$.
18. Обчислити суму арифметичної прогресії $3, 6, \ldots, 3n$, використовуючи функції +, – $(n > 0)$.
19. Обчислити суму геометричної прогресії $1, 3, 9, \ldots, 3^n$, використовуючи функції $\times$, +, – $(n > 0)$.

### Завдання

1. Написати SIPL-програму розв'язку сформульованої задачі.
2. Побудувати дерево її синтаксичного виводу.
3. Перевірити синтаксичну правильність програми.

## 1.2 ПРАКТИЧНА РОБОТА 2. Семантика програм. Семантичні терми. Побудова та обчислення семантичного терму програми

### Основні теоретичні відомості

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Формалізація простої мови програмування. Формальний опис семантики мови SIPL (Дані. Функції. Композиції. Програмні алгебри. Визначення семантичних термів. Побудова семантичного терму програми. Обчислення значень семантичних термів) (сторінка 12-21).

Процес побудови семантичного терму програми полягає в послідовному перетворенні запису, для якого будується семантичний терм. Ці перетворення вимагають таких дій:

- вибір загального правила, яке можна застосувати до підзапису поточного виразу з урахуванням пріоритету операцій;
- знаходження уніфікатора лівої частини правила з обраним підзаписом;
- отримання конкретного правила застосуванням уніфікатора до обох частин загального правила;
- заміна в поточному записі лівої частини конкретного правила на його праву частину.

Використовуватимемо формули для обчислення композицій і функцій алгебри $A_{Prog}$ (тут $f$ – $n$-арна функція, $fa, g_1, \ldots, g_n$ – номінативні арифметичні функції, $fb$ – номінативний предикат, $fs$, $fs1$, $fs2$ – біномінативні функції, $st$ – стан, $n$ – число).

Побудована алгебра $A_{SIPL}$ дозволяє тепер формалізувати семантику програм мови SIPL, задаючи їх функціональними виразами (семантичними термами) алгебри $A_{SIPL}$. Програма мови SIPL може бути перетворена на семантичний терм (терм програмної алгебри), який задає її семантику (семантичну функцію), перетвореннями такого типу:

- sem_P: Prog→ TFS;
- sem_S: Stm→ TFS;
- sem_A: Aexp→ TFA;
- sem_B: Bexp→ TFB.

| **Композиція** | **Формула обчислення** | **Ім'я формули** |
|---|---|---|
| Суперпозиція | $(S^n(f, g_1, \ldots, g_n))(st) = f(g_1(st), \ldots, g_n(st))$ | AF_S |
| Присвоювання | $AS^x(fa)(st) = st\nabla[x \mapsto fa(st)]$ | AF_AS |
| Послідовне виконання | $f_{S1}f_{S2}(st) = f_{S2}(f_{S1}(st))$ | AF_SEQ |
| Умовний оператор | $IF(f_b, f_{s_1}, f_{s_2})(st) = = \begin{cases} f_{s_1}(st), & \text{якщо } f_b(st) = \text{true}, \\ f_{s_2}(st), & \text{якщо } f_b(st) = \text{false}. \end{cases}$ | AF_IF |
| Цикл | $WH(f_b, f_S)(st) = st_n$, де $st_0 = st, st_1 = f_S(st_0)$, $st_2 = f_S(st_1), \ldots, st_n = f_S(st_{n-1})$, причому $f_b(st_0) = \text{true}, \ldots,$ $f_b(st_{n-1}) = \text{true}, f_b(st_n) = \text{false}.$ | AF_WH |
| Функція розіменування | $x\Rightarrow(st) = st(x)$ | AF_DNM |
| Тотожна функція | $id(st) = st$ | AF_ID |

Ці перетворення задаються рекурсивно (за структурою програми). Тому побудова семантичного терму залежить від вибору структури синтаксичного запису програми. Тут треба зважати на неоднозначність обраної нами граматики, що може зумовити різну семантику програм. Для досягнення однозначності треба користуватися пріоритетами операцій і типом їх асоціативності.

| **Правило заміни** | **Номер правила** |
|---|---|
| sem_P: Prog→TFS задається правилами:<br>sem_P( **begin** S **end**) = sem_S(S) | <br>NS_Prog |
| sem_S: Stm→TFS задається правилами:<br>sem_S(x := a) = $AS^x$(sem_A(a))<br>sem_S(S1; S2) = sem_S(S1) · sem_S(S2)<br>sem_S(**if** b **then** S1 **else** S2) =<br>IF(sem_B(b), sem_S(S1), sem_S(S2))<br>sem_S(**while** b **do** S) = WH(sem_B(b), sem_S(S))<br>sem_S(**begin** S **end**) = (sem_S(S))<br>sem_S(skip) = id | <br>NS_Stm_As<br>NS_Stm_Seq<br>NS_Stm_If<br>NS_Stm_Wh<br>NS_Stm_BE<br>NS_Stm_skip |
| sem_A: Aexp→TFA задається правилами:<br>sem_A(n) = $\overline{n}$<br>sem_A(x) = x⇒<br>sem_A(a1 + a2) = $S^2$(add, sem_A(a1), sem_A(a2))<br>sem_A(a1 - a2) = $S^2$(sub, sem_A(a1), sem_A(a2))<br>sem_A(a1 * a2) = $S^2$(mult, sem_A(a1), sem_A(a2))<br>sem_A(a) = sem_A(a) | <br>NS_A_Num<br>NS_A_Var<br>NS_A_Add<br>NS_A_Sub<br>NS_A_Mult<br>NS_A_Par |
| sem_B: Bexp→TFB задається правилами:<br>sem_B(a1 = a2) = $S^2$(eq, sem_A(a1), sem_A(a2))<br>sem_B(a1 > a2) = $S^2$(gr, sem_A(a1), sem_A(a2))<br>sem_B(b1 ∨ b2) = $S^2$(or, sem_B(b1), sem_B(b2))<br>sem_B(¬b) = $S^1$(neg, sem_B(b))<br>sem_B(b) = sem_B(b) | <br>NS_B_eq<br>NS_B_gr<br>NS_B_or<br>NS_B_neg<br>NS_B_Par |

Наведені правила слід розглядати як загальні правила, які в логіці називають *схемами правил*. Щоб із загального правила (метаправила) отримати *конкретне* (об'єктне) правило, слід замість синтаксичних метасимволів, таких як $a$, $b$, $S$, $P$, $n$, $x$, підставити конкретні синтаксичні елементи (записи), наприклад замість $a$ підставити $N - M$, замість $b$ – $M > N$ і т. д. Далі ліва частина конкретного правила замінюється на його праву частину і т. д.

Семантичні терми програми є точно (формально) заданими об'єктами, які формалізують семантику програм у термінах відповідних семантичних алгебр. Використовуватимемо найпростішу властивість термів, зважаючи на те, що вони задають деякі функції - *аплікацію*, що полягає в застосуванні функції, що задається термом, до певних вхідних даних. Аплікація є аналогом (абстракцією) тестування програм.

### Практичне завдання (приклад)

Побудувати семантичний терм для SIPL-програми для обчислення функції $x^y$, отриманої в попередньому розділі.

Для цього до тексту програми застосуємо правило заміни NS_Prog:

sem_P(begin R := 1; while Y > 0 do begin R := R * X; Y := Y - 1 end end) = sem_S(R := 1; while Y > 0 do begin R := R * X; Y := Y - 1 end).

Наступним потрібно застосовано правило NS_Stm_Seq: sem_S(R := 1; while Y > 0 do begin R := R * X; Y := Y - 1 end) = sem_S(R := 1) · sem_S(while Y > 0 do begin R := R * X; Y := Y - 1 end).

Звідки згідно з правилом NS_Stm_As: sem_S(R := 1) = $AS^R$(sem_A(1)). Де за правилом NS_A_Num: sem_A(1) = $\overline{1}$. Об'єднуючи отримані результати - маємо: sem_S(R := 1) = $AS^R(\overline{1})$.

В свою чергу для циклу за правилом NS_Stm_Wh: sem_S(while Y > 0 do begin R := R * X; Y := Y - 1 end) = WH(sem_B(Y > 0), sem_S(begin R := R * X; Y := Y - 1 end)). Розпишемо окремо терм, що відповідає умові циклу та тілу циклу. Застосовуючи послідовно правила NS_B_gr, NS_A_Var та NS_A_Num до умови циклу: sem_B(Y > 0) = $S^2$(gr, sem_A(Y), sem_A(0)) = $S^2$(gr, Y⇒, sem_A(0)) = $S^2$(gr, Y⇒, $\overline{0}$). Для тіла циклу потрібно спочатку застосувати правила NS_Stm_BE та NS_Stm_Seq: sem_S(begin R := R * X; Y := Y - 1 end) = sem_S(R := R * X; Y := Y - 1) = sem_S(R := R * X) · sem_S(Y := Y - 1). Перше присвоєння згідно правил NS_Stm_As, NS_A_Mult та NS_A_Var перетворюється на sem_S(R := R * X) = $AS^R$(sem_A(R * X)) = $AS^R$($S^2$(mult, sem_A(R), sem_A(X))) = $AS^R$($S^2$(mult, R⇒, X⇒)). Аналогічно для другого присвоєння, застосовуючи правила NS_Stm_As, NS_A_Mult, NS_A_Num та NS_A_Var отримуємо: sem_S(Y := Y - 1) = $AS^Y$(sem_A(Y - 1)) = $AS^Y$($S^2$(sub, sem_A(Y), sem_A(1))) = $AS^Y$($S^2$(sub, Y⇒, $\overline{1}$)).

Таким чином для циклу вірно наступне: sem_S(while Y > 0 do begin R := R * X; Y := Y - 1 end) = WH($S^2$(gr, Y⇒, $\overline{0}$), $AS^R$($S^2$(mult, R⇒, X⇒)) · $AS^Y$($S^2$(sub, Y⇒, $\overline{1}$))).

Підставляючи результати для присвоєння та циклу, отримуємо загальний терм для програми : $AS^R(\overline{1})$ · WH($S^2$(gr, Y⇒, $\overline{0}$), $AS^R$($S^2$(mult, R⇒, X⇒)) · $AS^Y$($S^2$(sub, Y⇒, $\overline{1}$))).

Проведемо тестування програми на вхідних даних $x = 5, y = 2$. Для цього застосуємо отриманий семантичний терм до стану $st = [X \mapsto 5, Y \mapsto 2]$.

Згідно формули AF_SEQ $AS^R(\overline{1})$ · WH($S^2$(gr, Y⇒, $\overline{0}$), $AS^R$($S^2$(mult, R⇒, X⇒)) · $AS^Y$($S^2$(sub, Y⇒, $\overline{1}$)))(st) = WH($S^2$(gr, Y⇒, $\overline{0}$), $AS^R$($S^2$(mult, R⇒, X⇒)) · $AS^Y$($S^2$(sub, Y⇒, $\overline{1}$)))($AS^R(\overline{1})$(st)).

Обчислимо окремо за формулою AF_AS $AS^R(\overline{1})$(st) = $st\nabla[R \mapsto \overline{1}(st)]$ = $[X \mapsto 5, Y \mapsto 2, R \mapsto 1] = st_0$

Залишається лише застосувати терм, що відповідає циклу до отриманого стану $st_0$. Для цього обчислимо умову циклу перед першою ітерацією:

$S^2(gr, Y\Rightarrow, \overline{0}(st_0)) = gr(Y\Rightarrow (st_0), \overline{0}(st_0)) = gr(2, 0) = true$. Умова циклу істинна, отже принаймні одна ітерація циклу має бути виконана. Стан після виконання даної ітерації позначимо $st_1$, цей стан може бути обчислений згідно формул:

$st_1 = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))\cdot AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(st_0) = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(AS^R(S^2(mult, R\Rightarrow, X\Rightarrow)))(st_0)$.

Окремо для кожного присвоєння маємо:

$st_0^1 = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))(st_0) = st_0\nabla[R \mapsto S^2(mult, R\Rightarrow, X\Rightarrow)(st_0)] = st_0\nabla[R \mapsto mult(R\Rightarrow (st_0), X\Rightarrow (st_0))] = st_0\nabla[R \mapsto mult(1, 5)] = st_0\nabla[R \mapsto 5] = [X \mapsto 5, Y \mapsto 2, R \mapsto 5]$.

$st_0^2 = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(st_0^1) = st_0^1\nabla[Y \mapsto S^2(sub, Y\Rightarrow, \overline{1})(st_0^1)] = st_0^1\nabla[Y \mapsto sub(2, 1)] = st_0^1\nabla[Y \mapsto 1] = [X \mapsto 5, Y \mapsto 1, R \mapsto 5] = st_1$.

Тут $st_0^1$ та $st_0^2$ позначають стани програми після виконання першого та другого присвоєння в тілі циклу. Отже, станом після виконання однієї ітерації циклу буде стан $st_1 = st_0^2$. Для нього і обчислимо умови продовження циклу.

$S^2(gr, Y\Rightarrow, \overline{0})(st_1) = gr(Y\Rightarrow (st_1), \overline{0}(st_1)) = gr(1, 0) = true$. Умова продовження циклу істинна, отже потрібно виконати додаткову ітерацію циклу:

$st_2 = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))\cdot AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(st_1) = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(AS^R(S^2(mult, R\Rightarrow, X\Rightarrow)))(st_1)$

$st_1^1 = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))(st_1) = st_1\nabla[R \mapsto S^2(mult, R\Rightarrow, X\Rightarrow)(st_1)] = st_1\nabla[R \mapsto mult(R\Rightarrow (st_1), X\Rightarrow (st_1))] = st_1\nabla[R \mapsto mult(5, 5)] = st_1\nabla[R \mapsto 25] = [X \mapsto 5, Y \mapsto 1, R \mapsto 25]$.

$st_1^2 = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(st_1^1) = st_1^1\nabla[Y \mapsto S^2(sub, Y\Rightarrow, \overline{1})(st_1^1)] = st_1^1\nabla[Y \mapsto sub(1, 1)] = st_1^1\nabla[Y \mapsto 0] = [X \mapsto 5, Y \mapsto 0, R \mapsto 25] = st_2$.

Перевіряємо умову на стані $st_2$ після другої ітерації циклу: $S^2(gr, Y\Rightarrow, \overline{0})(st_2) = gr(Y\Rightarrow (st_2), \overline{0}(st_2)) = gr(0, 0) = false$. Умова хибна, таким чином, стан $st_2$ є результатом обчислення циклу на стані $st_0$, а отже і результатом обчислення семантичного терму, що відповідає програмі на вхідних даних. В результуючому стані змінна R має значення 25, що співпадає з $5^2$, тому можна зробити висновок, що на вхідних даних програма працює коректно і тестування завершено успішно.

### Перелік задач

### Варіанти

1. Для обчислення $x - y$, використовуючи функцію $-1$ $(x, y > 0)$. Вхідні дані: $x = 8, y = 2$.
2. Для обчислення $\lfloor \lg n \rfloor$, використовуючи функції div, mod, +, – $(n > 0)$. Вхідні дані: $n = 17$.
3. Для обчислення $3^x$, використовуючи функції $\times$, +, – $(x > 0)$. Вхідні дані: $x = 3$.
4. Для обчислення $\lfloor \log_2 n \rfloor$, використовуючи функції div, mod, +, – $(n > 0)$. Вхідні дані: $n = 4$.

5. Для обчислення $(2n)!!$ $(n > 0)$. Вхідні дані: $n = 2$.
6. Для обчислення $(2n + 1)!!$ $(n > 0)$. Вхідні дані: $n = 2$.
7. Для обчислення суми арифметичної прогресії $2, 4, \ldots, 2n$, використовуючи функції $+$, $-$ $(n > 0)$. Вхідні дані: $n = 3$.
8. Для обчислення суми геометричної прогресії $1, 2, 4, \ldots, 2^n$, використовуючи функції $\times$, $+$, $-$ $(n > 0)$. Вхідні дані: $n = 3$.
9. Для обчислення $\lfloor \log_x y \rfloor$, використовуючи функції div, mod, $+$, $-$ $(x > 1, y > 0)$. Вхідні дані: $x = 3, y = 7$.
10. Для обчислення $C_x^y (x, y > 0)$. Вхідні дані: $x = 2, y = 1$.
11. Для обчислення $F_n$, $n$-того елементу ряду Фібоначчі, $(n > 0)$. Вхідні дані: $n = 3$.
12. Для перевірки парності числа $n$, за допомогою циклу, не використовуючи функції div, mod $(n > 0)$. Вхідні дані: $n = 5$.
13. Для перевірки непарності числа $n$, за допомогою циклу, не використовуючи функції div, mod $(n > 0)$. Вхідні дані: $n = 5$.
14. Для перевірки чи ділиться $x$ на $y$, використовуючи функції $+$, $-$ $(x, y > 0)$. Вхідні дані: $x = 9, y = 5$.
15. Для перевірки простоти числа $n$, використовуючи функції div, mod, $+$, $-$ $(n > 0)$. Вхідні дані: $n = 3$.
16. Для обчислення $\lfloor \sqrt{n} \rfloor$, використовуючи функції $\times$, $+$, $-$ $(n > 0)$. Вхідні дані: $n = 7$.
17. Для обчислення $5^x$, використовуючи функції $\times$, $+$, $-$ $(x > 0)$. Вхідні дані: $x = 3$.
18. Для обчислення суми арифметичної прогресії $3, 6, \ldots, 3n$, використовуючи функції $+$, $-$ $(n > 0)$. Вхідні дані: $n = 3$.
19. Для обчислення суми геометричної прогресії $1, 3, 9, \ldots, 3^n$, використовуючи функції $\times$, $+$, $-$ $(n > 0)$. Вхідні дані: $n = 3$.

**Завдання**

1. Побудувати семантичний терм програми.
2. Застосувати отриманий семантичний терм до вказаних вхідних даних (тестування).

## 1.3 ПРАКТИЧНА РОБОТА 3. Семантична коректність програм. Часткова та повна коректність програм. Доведення коректності програм

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Формалізація простої мови програмування. Властивості програмної алгебри (сторінка 22-28).

**Практичне завдання (приклад)**

Довести семантичну правильність програми побудованої в попередніх розділах для обчислення функції $x^y$.

Для цього покажемо, що для довільних початкових даних $x, y > 0$ програма завершуватиметься, та значенням змінної $R$ після завершення програми буде $x^y$. Формально це означає, що результат застосування семантичного терму, побудованного в попередньому розділі, до стану $st = [X \mapsto x, Y \mapsto y]$ $(x, y > 0)$ - визначений та рівний $st' = [X \mapsto x', Y \mapsto y', R \mapsto x^y]$ для деяких $x', y'$.

Згідно формули AF_SEQ застосувавши семантичний терм до $st$ маємо:

$AS^R(\overline{1})\cdot WH(S^2(gr, Y\Rightarrow, \overline{0}), AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))\cdot AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))))(st) = WH(S^2(gr, Y\Rightarrow, \overline{0}), AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))\cdot AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))))(AS^R(\overline{1})(st))$

Спочатку обчислимо внутрішнє присвоєння і позначимо відповідний стан $st_0 : AS^R(\overline{1})(st) = st\nabla[R \mapsto \overline{1}(st)] = [X \mapsto x, Y \mapsto y, R \mapsto 1] = st_0$

Для того, щоб застосувати оператор циклу до стану $st_0$ та довести семантичну коректність програми, спочатку доведемо методом математичної індукції наступне твердження: якщо згідно умови циклу могло бути виконано $k$ ітерацій циклу, то в результаті разі програма знаходиться в стані $st_k = [X \mapsto x, Y \mapsto y - k, R \mapsto x^k]$.

**База індукції:**

Підставимо $k = 0$, маємо $st_0 = [X \mapsto x, Y \mapsto y - 0, R \mapsto x^0]$. Данний стан співпадає з обчисленим раніше станом, в якому програма знаходиться до початку виконання циклу. Отже, база індукції виконується.

**Крок індукції:**

Нехай для $k > 0$ твердження виконується ($st_k = [X \mapsto x, Y \mapsto y - k, R \mapsto x^k]$), доведемо для $k + 1$. Для цього обчислимо $st_{k+1}$, застосувавши семантичний терм, що відповідає тілу циклу до стану $st_k$.

$st_{k+1} = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow)) \cdot AS^Y(S^2(sub, Y\Rightarrow, \overline{1})))(st_k) = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))(st_k))$

Аналогічно до тестування послідовно обчислимо обидва присвоєння: $st_k^1 = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))(st_k) = st_k\nabla[R \mapsto S^2(mult, R\Rightarrow, X\Rightarrow)(st_k)] = st_k\nabla[R \mapsto mult(x^k, x)] = [X \mapsto x, Y \mapsto y - k, R \mapsto x^{k+1}]$

$st_{k+1} = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(st_k^1) = st_k^1\nabla[Y \mapsto S^2(sub, Y\Rightarrow, \overline{1})(st_k^1)] = st_k^1\nabla[Y \mapsto sub(y - k, 1)] = [X \mapsto x, Y \mapsto y - (k + 1), R \mapsto x^{k+1}]$

Отримали шуканий стан, а отже, згідно з методом математичної індукції твердження виконується для довільного $k \geq 0$.

Дослідимо, чому буде рівна умова циклу для такого стану $st_k$:

$$S^2(gr, Y\Rightarrow, \overline{0})(st_k) = gr(Y\Rightarrow (st_k), 0) = gr(y-k, 0) = \begin{cases} true, \text{ якщо } k < y \\ false, \text{ якщо } k \geq y \end{cases}$$

Враховуючи визначення композиції циклу - умова має бути істинною після всіх ітерацій, що передують заключній, і бути хибною після заключної ітерації. Такій умові задовольняє саме $k = y$. Отжв заключний стан має вигляд: $st_y = [X \mapsto x, Y \mapsto y - y, R \mapsto x^y] = st'$. Що і доводить семантичну коректність програми, так як змінна $R$ має значення $x^y$. Завершуваність програми також випливає з дослідження значень умови циклу, так як гарантовано за $y$ ітерацій в змінній $Y$ буде значення не більше нуля, а отже цикл завершить свою роботу.

Розглянемо й інший підхід до доведення семантичної коректності програми, для цього позначимо $st^k$ - стан в якому перебуває програма за $k$ ітерацій до завершення роботи циклу. Значення змінних в такому стані позначимо відповідно $st^k = [X \mapsto x_k, Y \mapsto y_k, R \mapsto r_k]$

Доведемо методом математичної індукції, що для всіх $k \geq 0$ виконується $x_k = x_0; r_k * x_k^{y_k} = r_0 * x_0^{y_0}; y_k = y_0 + k$.

**База індукції:**

Підставимо $k = 0$: $x_0 = x_0; r_0 * x_0^{y_0} = r_0 * x_0^{y_0}; y_0 = y_0 + 0$. База очевидно виконується.

**Крок індукції:** Нехай для $k > 0$ твердження виконується, доведемо для $k + 1$. Так як стан $st^{k+1}$ це стан за $k + 1$ ітерацію до завершення роботи циклу, застосувавши до нього семантичний терм, що відповідає тілу циклу отримаємо стан $st^k$ (виконавши одну з $k + 1$ ітерації, залишається виконати $k$).

Виконавши відповідні перетворення отримуємо:
$AS^R(S^2(mult, R\Rightarrow, X\Rightarrow)) \cdot AS^Y(S^2(sub, Y\Rightarrow, \overline{1})))(st^{k+1}) =$
$AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))(st^{k+1}))$

$st_0^{k+1} = AS^R(S^2(mult, R\Rightarrow, X\Rightarrow))(st^{k+1}) = [X \mapsto x_{k+1}, Y \mapsto y_{k+1}, R \mapsto r_{k+1} * x_{k+1}]$

$st^k = AS^Y(S^2(sub, Y\Rightarrow, \overline{1}))(st_0^{k+1}) = [X \mapsto x_{k+1}, Y \mapsto y_{k+1} - 1, R \mapsto r_{k+1} * x_{k+1}]$

Об'єднуючи з позначеннями значень змінних в стані $st^k$: $st^k = [X \mapsto$

$x_k, Y \mapsto y_k, R \mapsto r_k] = [X \mapsto x_{k+1}, Y \mapsto y_{k+1} - 1, R \mapsto r_{k+1} * x_{k+1}]$. Звідки отримуємо наступні співвідношення змінних: $x_k = x_{k+1}; y_k = y_{k+1} - 1; r_k = r_{k+1} * x_{k+1}$, підставляючи припущення індукції маємо: $x_{k+1} = x_0; y_{k+1} = y_k + 1; y_{k+1} = y_0 + k + 1; r_{k+1} * x_{k+1}^{y_{k+1}} = r_{k+1} * x_{k+1}^{y_k+1} = r_{k+1} * x_{k+1} * x_{k+1}^{y_k} = r_k * x_k^{y_k} = r_0 * x_0^{y_0}$.

Припущення індукції доведено, а отже для довільного $k \geq 0$ виконується $x_k = x_0; r_k * x_k^{y_k} = r_0 * x_0^{y_0}; y_k = y_0 + k$.

Розглянемо при $k = n$ - стан до виконання першої ітерації циклу $x_n = x_0; r_n * x_n^{y_n} = r_0 * x_0^{y_0}$. Згідно обчисленого раніше $x_n = x, y_n = y, r_n = 1$, підставляючи в твердження доведене за допомогою математичної індукції $x = x_0; 1 * x^y = r_0 * x_0^{y_0}; y = y_0 + n$. Спрощуємо $x_0 = x; r_0 * x^{y_0} = x^y; y_0 = y - n$. Враховуючи умову завершення циклу:$S^2(gr, Y \Rightarrow, \overline{0})(st^0) = gr(Y \Rightarrow (St^0), 0) = gr(y_0, 0) = false$, маємо $y_0 \leq 0$. Випадок $y_0 < 0$ неможливий, бо згідно з доведеним тревдженням, тоді б $y_1 \leq 0$ і цикл би завершився раніше. Отже, $y_0 = 0$ і тому $r_0 * x^{y_0} = r_0 * x^0 = x^y$, звідки $r_0 = x^y$. Що і потрібно було довести - в змінній $R$ після завершення циклу міститься значення $x^y$, отже програма семантично правильно. Завершуваність програми доводить співвідношення $y = y_0 + n$ згідно з яким значення змінної $Y$ зменшується кожної ітерації, отже умова циклу через певну кількість ітерацій стане хибною.

### Перелік задач

### Завдання

Довести семантичну правильність програми (верифікація) побудованої при виконанні Практичної роботи 1-2.

## 1.4 МОДУЛЬНА КОНТРОЛЬНА РОБОТА 1. Перелік теоретичних питань та базових задач

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Формалізація простої мови програмування (сторінка 6-30).

**Теоретичні питання до контрольної роботи 1:**

1. Властивості основної пентади програмування.
2. Властивості програмної пентади.
3. Що таке часткова коректність програм? Яким чином доводиться часткова коректність програм?
4. Що таке повна коректність програм? Яким чином доводиться повна коректність програм?
5. Розкрийте зміст формалізації поняття програми. 6. Визначте різні класи функцій.
7. Визначте програмні системи різного рівня абстракції.
8. Дайте визначення класу номінативних даних.

**Типове завдання модульної контрольної роботи 1:**

1. Описати основні поняття програмування та їх відношення.
2. Побудувати програму на мові SIPL для однієї із наступних задач (x,y, n – додатні цілі числа):
- обчислення $x * y$, використовуючи функції +, –,
- обчислення $n!$,
- обчислення $x - y$, використовуючи функцію віднімання одиниці (–1),
- перевірки парності числа n,
- обчислення $x$ div $y$, використовуючи функції +, –,
- обчислення $x^y$, використовуючи функції *,+, –,
- обчислення $\lfloor \lg n \rfloor$, використовуючи функції div, mod, +, –,
- обчислення $x \mod y$, використовуючи функції +, –,
- обчислення $3 * x$, використовуючи функції *,+, –,
3. Перевірити синтаксичну правильність створеної програми, побудувавши їх дерева синтаксичного виводу. Побудувати семантичні терми цих програм, та застосувати їх до певних вхідних даних.
4. Довести часткову та повну коректність створеної програми.

**Приклад розв'язання практичного завдання модульної контрольної роботи 1**

**Практичне завдання:** Написати програму обчислення $x \div y$, використовуючи функції $+, -$ $(x, y > 0)$. Вхідні дані: $x = 7$, $y = 4$.

**1. SIPL-програма**

```
begin
    R := 0;
    while X >= 0 do
    begin
        X := X - Y;
        R := R + 1
    end
end
```

**2. Дерево синтаксичного виводу**

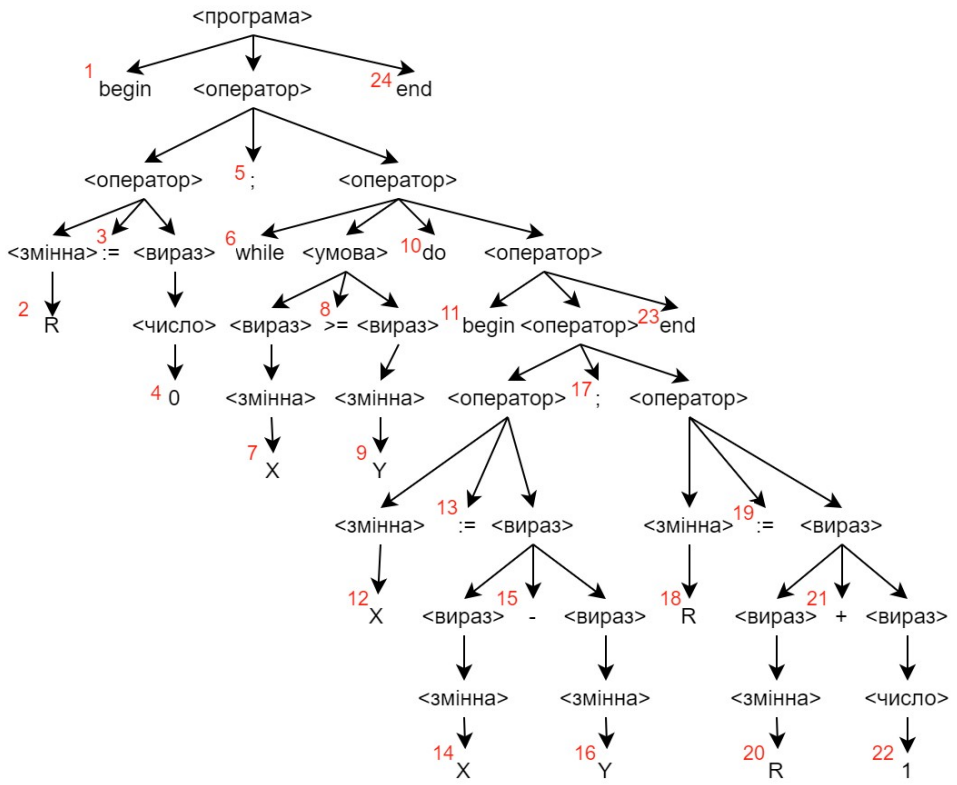


Рис. 2: Дерево синтаксичного виводу

**3. Перевірка синтаксичної правильності програми**

Пройшовши дерево виводу зліва на право, виписавши значення в листках (термінали), отримаємо begin R:=0; while X≥Y do begin X:=X-Y; R:=R+1 end end. Це співпадає з приведеною програмою, отже вона є синтаксично правильною.

**4. Семантичний терм програми**

$\text{Sem\_P}(\text{begin } R := 0;\ \text{while } X \geq Y \text{ do begin } X := X - Y; R := R + 1 \text{ end end}) =$
$\text{Sem\_S}(R := 0;\ \text{while } X \geq Y \text{ do begin } X := X - Y; R := R + 1 \text{ end}) =$
$\text{Sem\_S}(R := 0) \cdot \text{Sem\_S}(\text{while } X \geq Y \text{ do begin } X := X - Y; R := R + 1 \text{ end}) =$
$AS^R(\text{Sem\_A}(0)) \cdot \text{WH}(\text{Sem\_B}(X \geq Y),$
$\text{Sem\_S}(\text{begin } X := X - Y; R := R + 1 \text{ end})) =$
$AS^R(\overline{0}) \cdot \text{WH}(S^2(\text{gre}, \text{Sem\_A}(X), \text{Sem\_A}(Y)), \text{Sem\_S}(X := X - Y; R := R + 1)) =$
$AS^R(\overline{0}) \cdot \text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow), \text{Sem\_S}(X := X - Y) \cdot \text{Sem\_S}(R := R + 1)) =$
$AS^R(\overline{0}) \cdot \text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow), AS^X(\text{Sem\_A}(X - Y)) \cdot AS^R(\text{Sem\_A}(R + 1))) =$
$= AS^R(\overline{0}) \cdot \text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow),$
$AS^X(S^2(\text{sub}, \text{Sem\_A}(X), \text{Sem\_A}(Y))) \cdot AS^R(S^2(\text{add}, \text{Sem\_A}(R), \text{Sem\_A}(1)))) =$
$AS^R(\overline{0}) \cdot \text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow), AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow)) \cdot AS^R(S^2(\text{add}, R\Rightarrow, \overline{1})))$

**5. Застосувати отриманий семантичний терм до вказаних вхідних даних (тестування).**

$st = [X \mapsto 7, Y \mapsto 4]$

$AS^R(\overline{0}) \cdot \text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow), AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow)) \cdot AS^R(S^2(\text{add}, R\Rightarrow, \overline{1})))(st) =$

$\text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow), AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow)) \cdot AS^R(S^2(\text{add}, R\Rightarrow, \overline{1})))(AS^R(\overline{0})(st))$

$AS^R(\overline{0})(st) = st\nabla[R \mapsto \overline{0}(st)] = st\nabla[R \mapsto 0] = [X \mapsto 7, Y \mapsto 4, R \mapsto 0] = st'.$

Обчислимо:

$$\text{WH}(S^2(\text{gre}, X\Rightarrow, Y\Rightarrow), AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow)) \cdot AS^R(S^2(\text{add}, R\Rightarrow, \overline{1})))(st').$$

$$S^2(\text{gre}, X\Rightarrow, Y\Rightarrow)(st') = \text{gre}(X\Rightarrow(st'), Y\Rightarrow(st')) = \text{gre}(7, 4) = \text{true}.$$

Отже цикл виконується принаймні один раз:

$$st_1 = AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow)) \cdot AS^R(S^2(\text{add}, R\Rightarrow, \overline{1}))(st') =$$

$$AS^R(S^2(\text{add}, R\Rightarrow, \overline{1}))(AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow))(st'))$$

$$AS^X(S^2(\text{sub}, X\Rightarrow, Y\Rightarrow))(st') = st'\nabla[X \mapsto S^2(\text{sub}, X\Rightarrow, Y\Rightarrow)(st')] =$$

$$= st'\nabla[X \mapsto \text{sub}(X\Rightarrow(st'), Y\Rightarrow(st'))] = st'\nabla[X \mapsto \text{sub}(7, 4)] = st'\nabla[X \mapsto 3] =$$

$$= [X \mapsto 3, Y \mapsto 4, R \mapsto 0] = st''.$$

$$AS^R(S^2(\text{add}, R\Rightarrow, \overline{1}))(st'') = st''\nabla[R \mapsto S^2(\text{add}, R\Rightarrow, \overline{1})(st'')] =$$

$$= st''\nabla[R \mapsto \text{add}(R\Rightarrow(st''), \overline{1}(st''))] = st''\nabla[R \mapsto add(0, 1)] = st''\nabla[R \mapsto 1] =$$

$$= [X \mapsto 3, Y \mapsto 4, R \mapsto 1] = st_1.$$

Перевіряємо умову:

$$S^2(\text{gre}, X\Rightarrow, Y\Rightarrow)(st_1) = \text{gre}(X\Rightarrow (st_1), Y\Rightarrow (st_1)) = \text{gre}(7,4) = \text{false}.$$

Таким чином, маємо результат:

$$st_1 = [X \mapsto 3, Y \mapsto 4, R \mapsto 1].$$

Тут в змінній $R$ записано результат операції $x \div y$ для значень $x = 7$, $y = 4$, отже протестована функція вірна на вхідних даних.

**6. Довести семантичну правильність програми (верифікація)**

Нехай вхідні данні $st = [X \mapsto x, Y \mapsto y]$. Тоді якщо виконаються $k$ ітерацій циклу ми отримаємо стан $st_k = [X \mapsto x - ky, Y \mapsto y, R \mapsto k]$.

**База індукції:**

$st_0 = [X \mapsto x, Y \mapsto y, R \mapsto 0]$. Виконується, бо $AS^R(\bar{0})(st) = st\nabla[R \mapsto (st)] = st\nabla[R \mapsto 0] = [X \mapsto x, Y \mapsto y, R \mapsto 0]$.

**Крок індукції:**

Припустимо, що для $k$ це вірно, доведемо для $k+1$.

Для цього застосуємо тіло циклу до стану $st_k = [X \mapsto x - ky, Y \mapsto y, R \mapsto k]$.

$st_{k+1} = AS^X(S^2(\text{sub}, X \Rightarrow, Y \Rightarrow)) \cdot AS^R(S^2(\text{add}, R \Rightarrow, \bar{1}))(st_k) = AS^R(S^2(\text{add}, R \Rightarrow, \bar{1}))(AS^X(S^2(\text{sub}, X \Rightarrow, Y \Rightarrow))(st_k))$

$AS^X(S^2(\text{sub}, X \Rightarrow, Y \Rightarrow))(st_k) = st_k\nabla[X \mapsto S^2(\text{sub}, X \Rightarrow, Y \Rightarrow)(st_k)] = st_k\nabla[X \mapsto \text{sub}(X \Rightarrow st_k), Y\Rightarrow (st_k))] = st_k\nabla[X \mapsto sub(x-ky, y)] = st_k\nabla[X \mapsto x - ky - y] = [X \mapsto x - (k+1)y, Y \mapsto y, R \mapsto k] = st_k$

$AS^R(S^2(\text{add}, R \Rightarrow, \bar{1}))(st'_k) = st'_k\nabla[R \mapsto S^2(\text{add}, R \Rightarrow, \bar{1})(st'_k)] = st'_k\nabla[R \mapsto \text{add}(R \Rightarrow (st'_k), \bar{1}(st_k`)) = st_k`\nabla = [R \mapsto add(k,1)] = st``\nabla[R \mapsto k+1] = [X \mapsto x - (k+1)y, Y \mapsto y, R \mapsto k+1] = st_{k+1}$

Отримали те, що потрібно, отже твердження індукції виконується.

З доведеного твердження ми маємо, що якщо цикл завершився після $k$ ітерацій, то результуючий стан буде $st_k = [X \mapsto x - ky, Y \mapsto y, R \mapsto k]$. З умови виходу з циклу, а саме $S^2(\text{gre}, X \Rightarrow, Y\Rightarrow)(st_{k-1}) = \text{gre}(X \Rightarrow (st_k), Y\Rightarrow (st_k)) = \text{gre}(x-ky, y) = \text{false}$ маємо, що $x-ky < y$. Так як цикл на кроці $k-1$ ще виконувався, маємо: $S^2(\text{gre}, X \Rightarrow, Y \Rightarrow)(st_{k-1}) = \text{gre}(X \Rightarrow (st_{k-1}), Y \Rightarrow (st_{k-1})) = \text{gre}(x - (k-1)y, y) = \text{true}$, отже $x - (k-1)y \geq y$. Що доводить $x \div y = k$. Отже, дійсно якщо цикл після $k$ ітерацій завершується, для деякого $k$, результат операції $x \div y$ буде записано в змінній $R$, що й треба було довести.

# 2 ФОРМАЛЬНІ МОВИ І ГРАМАТИКИ

## 2.1 ПРАКТИЧНА РОБОТА 4. Класифікація мов та граматик. Нормальні форми

### Основні теоретичні відомості

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf
Синтактика: формальні мови та граматики.
Визначення основних понять формальних мов (сторінка 66-70).
Породжуючі граматики (сторінка 60-75).
Властивості контекстно-вільних граматик (Видалення несуттєвих символів. Видалення $\varepsilon$-правил. Нормальна форма Хомського. Нормальна форма Грейбах) (сторінка 86-89).

Наведемо властивості контекстно-вільних граматик (КВ-граматик), необхідні для ппобудови нормальних форм заданих граматик, зокрема їхні еквівалентні перетворення та нормальні форми.

***1. Перейменування нетерміналів.***

Лема (про перейменування нетерміналів). Нехай задані граматика $G = (N, T, P, S)$ і бієктивне відображення $\rho : N \to N'$ множини нетерміналів $N$ на деяку іншу множину нетерміналів $N'$ $(N' \cap T = \emptyset)$. Тоді для граматики $G' = (N', T, S', P')$, де $P'$ – множина правил, отриманих із $P$ заміною нетерміналів N на відповідні нетермінали з $N'$, а $S' = \rho(S)$, маємо: $L(G) = L(G')$.

***2. Видалення несуттєвих символів.***

Нетермінал $A \in N \cup T$ назвемо недосяжним у граматиці $G = (N, T, P, S)$, якщо $A$ не з'являється в жодному вивідному ланцюжку, тобто не існує виведення вигляду $S \Rightarrow^*_G \gamma A \delta, \gamma, \delta \in (N \cup T)^*$. Для знаходження недосяжних нетерміналів спочатку визначимо множину досяжних нетерміналів. Ця множина для заданої КВ-граматики $G = (N, T, P, S)$ легко визначається за допомогою таких індуктивних визначень:

1. $R_0 = \{S\}$.

2. $R_i = \{B \mid$ є правило $A \to \alpha B \beta \in P$, що $A \in R_{i-1}, B \in N\} \cup R_{i-1}$ $(i = 1, 2, \dots)$.

Оскільки множина $N$ є скінченною, а формула для визначення $R_i$ задає монотонне за $i$ відображення, то існує $k (k \geq 0)$ таке, що $R_k = R_{k+1}$. Інакше кажучи, послідовність $R_0, R_1, R_2, \dots$ стабілізується на k-му кроці. Покладемо $R = R_k$. Множина $UR$ недосяжних нетерміналів задається формулою $UR = N \backslash R$.

За граматикою $G = (N, T, P, S)$ будуємо граматику $G' = (N', T', P', S')$ таким чином:

1. $N' = N \cap R$.
2. $T' = T$.
3. $S' = S$.
4. $P' = \{A \to \alpha \mid \alpha \in (R \cup T)^*\}$.

Лема. Для граматики $G' = (N', T', P', S')$ виконуються такі властивості:

1. $L(G') = L(G)$ (еквівалентність).

2. Для всіх $A \in N'$ існують такі ланцюжки $\alpha, \beta \in (N' \cup T)^*$, що $S \Rightarrow_G^* \alpha A \beta$ (усі нетермінали є досяжними).

Означення. Нетермінал $A \in N$ назвемо непродуктивним у граматиці $G = (N, T, P, S)$, якщо з $A$ не можна вивести жодного термінального ланцюжка, тобто не існує виведення вигляду $A \Rightarrow_G^* t, t \in T^*$.

Спочатку визначимо множину продуктивних нетерміналів. Множина продуктивних нетерміналів для заданої КВ-граматики $G = (N, T, P, S)$ легко визначається за допомогою таких індуктивних визначень:

1. $P_{r0} = \{A \mid$ є правило $A \to t \in$ , що $t \in T^*\}$.

2. $P_{ri} = \{A \mid$ є правило $A \to \gamma \in$ , що $\gamma \in (R_{i-1} \cup )^*\} \cup R_{i-1}, (i = 1, 2, \dots)$.

Оскільки множина $N$ є скінченною, а формула для визначення $P_{ri}$ задає монотонне за $i$ відображення, то існує $k (k \geq 0)$ таке, що $P_{r_k} = P_{r_k+1}$. Інакше кажучи, послідовність $P_{r_0}, P_{r_1}, P_{r_2}, \dots$ стабілізується на k-му кроці. Покладемо $P_r = P_{r_k}$.

Множина $UPr$ непродуктивних нетерміналів задається формулою $UPr = N\ Pr$. За граматикою $G = (N, T, P, S)$ будуємо граматику $G' = (N', T', P', S')$ таким чином:

1. $N' = (N' \cap Pr) \cup S$.
2. $T' = T$.
3. $S' = S$.
4. $P' = A \to \alpha P \mid \alpha \in (Pr \cup T)^*$.

Лема. Для граматики $G' = (N', T', P', S')$ виконуються такі властивості:

1.$L(G') = L(G)$ (еквівалентність).

2. Для всіх $A \in N' \setminus \{S\}$ існують термінальні ланцюжки, що виводяться з $A$.

Зазначимо, що аксіома $S$ є спеціальним випадком, тому що вона може бути непродуктивною, і разом з тим вона має належати множині нетермінальних символів (за визначенням породжувальних граматик), але в такому випадку множина правил буде порожньою.

Наведений метод побудови множини продуктивних нетерміналів дозволяє дати відповідь на запитання, чи є порожньою мова, що породжується граматикою $G = (N, T, P, S)$? Відповідь: якщо аксіома є продуктивним нетерміналом, то мова непорожня, якщо ж аксіома – непродуктивний нетермінал, то

мова – порожня.

Означення. Символ $X \in N \cup T$ назвемо несуттєвим у КВ-граматиці $G = (N, , P, S)$, якщо в ній немає виведення вигляду $S \Rightarrow^* wXe \Rightarrow^* wxy$, де $w, x, y \in T^*$.

З наведеного означення бачимо, що нетермінал – несуттєвий, якщо є недосяжним або непродуктивним нетерміналом (крім, можливо, аксіоми). Термінальний символ є несуттєвим, коли немає жодного породжуваного ланцюжка, що містить .

Означення. Граматика $G = (N, T, P, S)$ називається зведеною, якщо в ній немає несуттєвих символів (можливо, крім аксіоми $S$).

Для побудови зведеної граматики спочатку видаляємо недосяжні й непродуктивні нетермінали. Потім із множини термінальних символів видаляємо несуттєві символи. Такі термінальні символи не містяться в правих частинах отриманих правил. Отримана граматика буде зведеною.

Лема 5. За кожною КВ-граматикою можна побудувати еквівалентну їй зведену граматику, що не містить несуттєвих символів.

***3. Видалення $\varepsilon$-правил.***

Означення. Назвемо КВ-граматику $G = (N, T, P, S)$ граматикою без $\varepsilon$-правил (або нескорочуваною), якщо не містить $\varepsilon$-правил, тобто правил вигляду $A \to \varepsilon$.

Лема. За кожною КВ-граматикою можна побудувати еквівалентну їй (з точністю до порожнього ланцюжка – $\varepsilon$-еквівалентну) граматику без $\varepsilon$-правил.

Доведення. Нехай дана КВ-граматика $G = (N, T, P, S)$, що породжує мову $L(G)$. Проведемо серію перетворень множини $P$. Якщо множина $P$ містить правила $B \to \alpha A \beta$ і $A \to \varepsilon$ для деяких $A, B \in N, \alpha, \beta \in (N \cup T)^*$, то додамо правило $B \to \alpha\beta P$. Повторюємо цю процедуру, поки множина правил не стабілізується. Далі виключимо з множини $P$ усі правила вигляду $A \to \varepsilon$.

Наведений алгоритм можна уточнити за допомогою таких індуктивних визначень.

1. $P_0 = P$

2. $P_i = \{B \to \alpha\beta \mid$ є правила $B \to \alpha A \beta \in P, A \to \in P_{i-1}\} \cup R_{i-1} (i = 1, 2, \dots)$.

Оскільки множина правил є скінченною, а праві частини нових правил коротші, то формула для визначення $P_i$ задає монотонне за $i$ відображення. Тому існує $k (k \geq 0)$ таке, що $P_k = P_{k+1}$. (Послідовність $P_0, P_1, P_2, \dots$ стабілізується на k-му кроці). Покладемо $P' = P_k \setminus \{A \to \varepsilon \in P_k\}$.

Одержана граматика $G' = (N, T, P', S)$ породжує мову $L(G) \setminus \{\varepsilon\}$.

Дійсно, кожне виведення в граматиці $G$, яке не породжує порожнє слово, може бути промодельоване в граматиці $G'$, що легко доводиться індукцією за довжиною виведення. Так само можна промоделювати виведення в $G'$ виведеннями в $G$, використовуючи замість модифікованих правил вигляду $B \to \alpha\beta$

початкові правила вигляду $B \to \alpha A \beta$ з подальшим виведенням порожнього ланцюжка з $A$.

***4. Нормальна форма Хомсъкого.***

Означення. Назвемо КВ-граматику $G = (N, T, P, S)$ граматикою у нормальній формі Хомського, якщо її правила мають вигляд $S \to \varepsilon, A \to a, A \to BC$ для деяких $A, B, C \in N, a \in T$.

Лема. За кожною КВ-граматикою можна побудувати еквівалентну їй граматику в нормальній формі Хомського.

Доведення. Нехай дано КВ-граматику $G = (N, T, P, S)$. Проведемо низку перетворень цієї граматики так, що породжувана нею мова залишиться незмінною. Спочатку побудуємо еквівалентну граматику $G' = (N', T', P', S')$ без $\varepsilon$ -правил.

Далі замінимо в усіх правилах кожен термінальний символ a на новий нетермінальний символ $N(a)$ і додамо до множини $P$ правила $N(a) \to a$ для всіх $a \in T$.

Видалимо правила вигляду $A \to A_1 A_2 \dots A_n$, де $n > 2$, замінюючи їх на таку низку нових правил: $A \to A_1 N(A_2 \dots A_n)$, $N(A_2 \dots A_n) \to A_2 N(A_3 \dots A_n)$, $\dots$, $N(A_{n-1} \dots A_n) \to A_{n-1} A_n$ (при цьому додаються нові нетермінальні символи $N(A_2 \dots A_n), N(A_3 \dots A_n), \dots, N(A_{n-1} \dots A_n)$).

Якщо для деяких $A \in N, B$ і $\alpha \in (N \cup T)^*$ множина містить правила $A \to B, B \to \alpha$, але не містить правила $A \to \alpha$, то додамо це правило в $P$. Повторюємо процедуру доки можливо. Після цього видалимо з множини $P$ усі правила вигляду $A \to B$.

Нарешті, змінимо $S$ на $S'$ і додамо правила $S' \to \varepsilon, S \to S'$ у тому випадку, коли мова $L(G)$ містить порожній ланцюжок.

***5. Нормальна форма Грейбах.***

Означення. КВ-граматику $G = (N, T, P, S)$ будемо називати граматикою в нормальній формі Грейбах, якщо її правила мають вигляд $A \to a\alpha$ ($a \in T$, $\alpha \in (N \cup T)^*$), тобто кожне правило починається з термінального символу.

Лема. За кожною КВ-граматикою можна побудувати $\varepsilon$-еквівалентну їй (з точністю до порожнього ланцюжка) граматику в нормальній формі Грейбах.

**Практичне завдання (приклад)**

Для граматики: $S \to AbCa|ABB|BScB|aSB$, $A \to Aa|\varepsilon$, $B \to bB|\varepsilon$, $C \to Dc|Cc|c$, $D \to Dd$ побудувати нормальну форму без $\varepsilon$-правил.

***Розв'язування.*** Спочатку видалимо несуттєві нетермінали: непродуктивні та недосяжні. Для цього знайдемо множину всіх досяжних та продуктивних нетерміналів.

Множина досяжних за 0 кроків нетерміналів $R_0$ складається лише з початкового нетерміналу $S$. Множина $R_{i+1}$ - досяжних за не більше ніж $i + 1$ крок нетерміналів складається з усіх нетерміналів $N' \in R_i$, що досяжні за не

більше ніж $i$ кроків, а також усіх нетерміналів, яких можна досягти за один крок з таких нетерміналів $N'$. Тобто таких нетерміналів $N''$, що граматика містить правило $N' \to \alpha N'' \beta$.

$R_0 = \{S\}$. В правих частинах правил для нетерміналу $S$ містяться нетермінали $A, C, B$, а отже $R_1 = \{S, A, C, B\}$. З нетерміналу $A$ за один крок досяжний лише нетермінал $A$. З нетерміналу $B$ - $B$, а з нетерміналу $C$ - відповідно нетермінали $D$ та $C$. Отже $R_2 = R_1 \cup \{D\} = \{S, A, C, B, D\}$. $R_2$ співпалає з множиною всіх нетерміналів, присутніх в граматиці, тому алгоритм пошуку досяжних нетерміналів зупиняється і множина недосяжних нетерміналів - порожня.

Побудуємо ітеративно множину продуктивних нетерміналів. На першому кроці в множину $P_0$ додамо всі нетермінали, які мають правила, права частина яких складається лише з термінальних символів, або епсилон. В нашому випадку - це правила $A \to \varepsilon$, $B \to \varepsilon$, $C \to c$. Отже $P_0 = \{A, B, C\}$. На наступному кроці додамо нетермінали, що не належать множині $P_0$, проте мають правило, права частина якого що складається лише з термінальних символів та елементів $P_0$. Таким є правило $S \to AbCa$, тому $S$ є продуктивним. Для $D$ таке правило наразі відсутнє. Отже $P_1 = \{A, B, C, S\}$. Наступна ітерація не додасть нових нетерміналів, так як жодне правило для $D$ не складається лище з терміналів та нетерміналів, що були визначені на попередніх кроках, як продуктивні, таким чином продуктивними є лише нетермінали $A, B, C, S$.

Нетермінал $D$ є непродуктивним, тому він може бути видалений з граматики разом з відповідними йому правилами, та правилами в яких він міститься в правій частині.

Еквівалентна граматика без $D$ має вигляд: $S \to AbCa|ABB|BScB|aSB$, $A \to Aa|\varepsilon$, $B \to bB|\varepsilon$, $C \to Cc|c$. Для неї застосуємо наступні кроки алгоритму побудови нормальної форми без $\varepsilon$-правил.

Розглянемо перший $\varepsilon$-нетермінал $A$. Додамо нові правила, отримані з правил граматики, шляхом видалення входжень нетерміналу $A$ в правій частині: $S \to bCa|BB$, $A \to a$.

В результаті, можна видалити з граматики правило ~~$A \to \varepsilon$~~, отримавши еквівалектну граматику.

Розглянемо наступний $\varepsilon$-нетермінал $B$. В даном випадку, також враховуємо і правила отримані на попередньому кроці, видаляючи нетермінал $A$. Отримаємо нові правила: $S \to AB|A|BSc|ScB|Sc|aS$, $B \to b$, $S \to B|\varepsilon$

Варто зазначити, що для таких правил, як $S \to BScB$, потрібно розглянути всі можливі комбінації вилучення входжень нетерміналу $B$ в правій частині. Таким чином, потрібно додати декілька правил - $S \to ScB|BSc|Sc$.

Видаляємо з граматики правило ~~$B \to \varepsilon$~~

Можна побачити, що в результаті додавання нових правил, будо додано правило $S \to \varepsilon$, а отже $S$ є $\varepsilon$-нетерміналом і має також бути розглянутий. В результаті для нетерміналу $S$ маємо додати правила. $S \to BcB|aB$ $S \to Bc|cB|c|a$

Що дозволяє видалити правило ~~$S \to \varepsilon$~~

Об'єднуючи правила граматики, нові правила та видаляючи $\varepsilon$-правила, отримуємо нормальну форму без $\varepsilon$-правил:

$S \to AbCa|ABB|BScB|aSB|bCa|BB|AB|A|BSc|ScB|Sc|aS|B|BcB|aB$
$|Bc|cB|c|a$
$A \to Aa|a$
$B \to bB|b$
$C \to Cc|c$

Базуючись на побудованій нормальній формі без $\varepsilon$-правил, побудуємо нормальну форму за Хомським.

На першому кроці - введемо нові нетермінали для термінальних символів $a, b, c$: $A' \to a$, $B' \to b$, $C' \to c$

Замінюємюючи входження терміналів $a, b, c$ в правилах на відповідні нетермінали $A', B', C'$, отримаємо граматику:

$S \to AB'CA'|ABB|BSC'B|A'SB|B'CA'|BB|AB|A|BSC'|SC'B|SC'|A'S$
$|B|BC'B|A'B|BC'|C'B|c|a$
$A \to AA'|a$
$B \to B'B|b$
$C \to CC'|c$

Правила, що містять в правій частині менше, або більше дво нетерміналів необхідно замінити:

$S \to AB'CA'|ABB|BSC'B|A'SB|B'CA'|A|BSC'|SC'B|B|BC'B$

Для видалення правил $S \to A, S \to B$, що містять лише один нетермінал в правій частині, потрібно для нетерміналу $S$ додати всі правила для нетерміналів $A$ та $B$. Враховуючи, що правило $S \to a$ вже присутнє, це наступні правила: $S \to AA'|B'B|b$

Останнім кроком побудови нормальної форми за Хомським залишається замінити правила, що мають більше ніж два нетермінали в правій частині:

$S \to AB'CA'|ABB|BSC'B|A'SB|B'CA'|BSC'|SC'B|BC'B$

Для того, щоб видалити правило ~~$S \to AB'CA'$~~, введено нетермінал $D$ та правила $S \to AD$ та $D \to B'CA'$. Аналогічно, щоб видалити ~~$D \to B'CA'$~~, додаємо $D \to B'E$ та $E \to CA'$.

Для правила ~~$S \to ABB$~~, маємо нові правила $S \to AF$ та $F \to BB$.

Аналогічно для ~~$S \to BSC'B$~~, додаємо $S \to BG$ та ~~$G \to SC'B$~~, звідки маємо $G \to SH$ та $H \to C'B$.

Продовжуючи для ~~$S \to A'SB$~~ - потрібно ввести правила $S \to A'I$ та $I \to SB$. А для ~~$S \to B'CA'$~~ - правила $S \to B'J$ та $J \to CA'$. Також для решти правил, що залишилися - маємо для ~~$S \to BSC'$~~ - правила $S \to BK$ та $K \to SC'$, для ~~$S \to SC'B$~~ - правила $S \to SL$ та $L \to C'B$.

Так як правило $H \to C'B$ вже було додано раніше. Щоб видалити ~~$S \to BC'B$~~, достатньо додати лише $S \to BH$.

Об'єднуючи всі додані правила, маємо нормальну форму за Хомським:
$S \to BB|AB|SC'|A'S|A'B|BC'|C'B|c|a|AA'|B'B|b|AD|AF|BG|A'I|B'J|BK$
$|SL|BH$
$A \to AA'|a$
$B \to B'B|b$
$C \to CC'|c$
$D \to B'E$, $E \to CA'$, $F \to BB$, $G \to SH$, $H \to C'B$, $I \to SB$, $J \to CA'$
$K \to SC'$, $L \to C'B$
$A' \to a$, $B' \to b$, $C' \to c$

**Перелік задач** :

**Завдання.** Побудувати нормальну форму без $\varepsilon$-правил та нормальну форму Хомського для КВ-граматики:
$S \to AbBE$, $S \to ADcBE$, $S \to AEcC$, $E \to \varepsilon \mid bEc$, $A \to \varepsilon \mid aA$, $B \to \varepsilon \mid bB$, $C \to \varepsilon \mid cC$, $D \to d \mid dD$.

**Варіанти:**

1. $S \to bBA$, $S \to DcBA$, $S \to DAAC$, $S \to AcC$
   $A \to \varepsilon \mid bAc$, $B \to \varepsilon \mid bB$, $C \to \varepsilon \mid cC$, $D \to dD$.
2. $S \to AbB$, $S \to ADcB$, $S \to ADBB$, $S \to ABcC$
   $B \to \varepsilon \mid bBc$, $A \to \varepsilon \mid aA$, $C \to c \mid cC$, $D \to dD$.
3. $S \to AbBC$, $S \to ADcBC$, $S \to ADCC$, $S \to ACc$
   $C \to \varepsilon \mid bCc$, $A \to \varepsilon \mid aA$, $B \to \varepsilon \mid bB$, $D \to dD$.
4. $S \to AbBD$, $S \to AcBD$, $S \to ADdD$, $S \to ADcC$
   $D \to \varepsilon \mid bDc$, $A \to \varepsilon \mid aA$, $B \to \varepsilon \mid bB$, $C \to cC$.
5. $S \to EbBA$, $S \to EDcBA$, $S \to EDAA$, $S \to EAcC$
   $A \to \varepsilon \mid bAc$, $E \to \varepsilon \mid EE$, $B \to \varepsilon \mid bB$, $C \to \varepsilon \mid cC$, $D \to dD$.
6. $S \to aSbA$, $S \to ABCD$, $S \to DCBA$, $S \to BBcC$
   $A \to \varepsilon \mid aA$, $B \to bB$, $C \to \varepsilon \mid cC$, $D \to d \mid dD$.
7. $S \to BbB$, $S \to cCc$, $S \to AaA$, $S \to dDd$
   $A \to aA$, $B \to \varepsilon \mid bB$, $C \to \varepsilon \mid cC$, $D \to d \mid dD$.

## 2.2 ПРАКТИЧНА РОБОТА 5. Контекстно-вільні граматики. Рівняння в алгебрах формальних мов. Побудова мови за допомогою системи рівнянь з регулярними коефіцієнтами

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Синтактика: формальні мови та граматики.
Рівняння в алгебрах формальних мов (сторінка 95-96).

Граматики та БНФ є формалізмами, що задають мову поштучно, поелементно, указуючи механізм породження окремих слів. Разом з тим сама форма граматик і БНФ підштовхує до думки, що їх можна розглядати як системи рівнянь, розв'язками яких є мови. Щоб перейти до такого тлумачення, попередньо введемо необхідні означення.

**Означення.** Слабкою алгеброю формальних мов над алфавітом $T$ будемо називати алгебру $AL = (2^{T^*}, \cup, \cdot)$ з операціями об'єднання й добутку мов.

**Означення.** Множиною виразів $LExp$ над множиною змінних $Var$ і множиною мов-констант $LS = \{L_i \mid L_i \subseteq T^*, i \in I\}$ називається множина, задана таким індуктивним визначенням:

1. Якщо $x \in Var$, то $x \in LExp$;
2. Якщо $L \in LS$, то $L \in LExp$;
3. Якщо $t, t' \in LExp$, то $t \cup t'$, $t \cdot t' \in LExp$.

**Означення.** Формальним рівнянням (формальною рівністю) над алгеброю $AL$ називається запис вигляду $t = t'$, де $t, t' \in LExp$.

**Означення.** Інтерпретацією (означуванням, оцінкою) змінних називається довільне відображення $\nu : Var \to 2^{T^*}$. Інтерпретація змінних однозначно (і гомоморфно) продовжується до відображення $\mu_\nu : LExp \to 2^{T^*}$ інтерпретації виразів, параметром якого є інтерпретація змінних $\nu$, що задається індуктивно за побудовою виразу $e \in LExp$:

1. Якщо $e = x$ $(x \in Var)$, то $\mu_\nu(e) = \nu(x)$;
2. Якщо $e = L$ $(L \in LS)$, то $\mu_\nu(e) = L$;
3. Якщо $e = t \cup t'$, то $\mu_\nu(e) = \mu_\nu(t) \cup \mu_\nu(t')$;
4. Якщо $e = t \cdot t'$, то $\mu_\nu(e) = \mu_\nu(t) \cdot \mu_\nu(t')$.

**Означення.** Наведене визначення дозволяє абстрагувати відображення $\mu_\nu$ до оператора $\mu : LExp \to ((Var \to 2^{T^*}) \to 2^{T^*})$. Вираз $e$, який містить

змінні $x_1, \ldots, x_n$, будемо позначати $e(x_1, \ldots, x_n)$, а відповідний оператор – $\mu(e(x_1, \ldots, x_n))$.

**Приклад.** Нехай $Var = \{x, y, z\}$, $\nu(x) = \{a, ab\}$, $\nu(y) = \{\varepsilon, b, cc\}$, $\nu(z) = \{bb, c\}$. Тоді вираз $x^2z\{a\}y$ інтерпретується таким чином:
$\mu_\nu(x^2z\{a\}y) =$
$= \{a, ab\}^2\{bb, c\}\{a\}\{\varepsilon, b, cc\} =$
$= \{aa, aab, aba, abab\}\{bba, ca\}\{\varepsilon, b, cc\} =$
$= \{aabba, aabbba, ababba, ababbba, aaca, aabca, abaca, ababca\}\{\varepsilon, b, cc\} =$
$= \{aabba, aabbba, ababba, ababbba, aaca, aabca, abaca, ababca, b, cc\} =$
$= \{aabbab, aabbbab, ababbab, ababbbab, aacab, aabcab, abacab, ababcab, aabbacc,$
$aabbbacc, ababbacc, ababbbacc, aacacc, aabcacc, abacacc, ababcacc\}$.

**Означення.** Інтерпретація (означування) змінних $\nu : Var \to 2^{T^*}$ називається розв'язком рівняння $t = t'$, якщо $\mu_\nu(t) = \mu_\nu(t')$.

**Означення.** Розв'язком системи рівнянь $\{t_1 = t'_1, \ldots, t_n = t'_n\}$ є така інтерпретація змінних, яка кожне рівняння перетворює на рівність.

**Означення.** Рівняння вигляду $x = t$, де $x \in Var$, $t \in LExp$, називається *рекурсивним*. Якщо у виразі $t$ фігурують лише скінченні мови, то рівняння називають *слабкорекурсивним*. Аналогічно вводяться поняття рекурсивних і слабкорекурсивних систем рівнянь.

**Приклад.** Розглянемо мову $L = \{a^n b^n \mid n \geq 0\}$, яка породжується простою граматикою $G : S \to \varepsilon \mid aSb$. Цій граматиці відповідає рівняння $S = \{\varepsilon\} \cup \{a\}S\{b\}$. Позначимо оператор $\mu(\{\varepsilon\} \cup \{a\}S\{b\})$ як $\varphi(S)$.

Розв'язок рівняння знаходимо методом послідовних наближень. Найперше наближення – порожня мова $\varepsilon$. Наступні наближення отримуємо підстановкою в оператор $\varphi(S)$ замість $S$ попереднього наближення(для спрощення тут можна говорити про підставку в саме рівняння). Отримуємо таку послідовність наближень:

$$\begin{aligned}
R^{(0)} &= \varnothing; \\
R^{(1)} &= \{\varepsilon\}; \\
R^{(2)} &= \{\varepsilon\} \cup \{ab\} = \{\varepsilon, ab\}; \\
R^{(3)} &= \{\varepsilon, ab\} \cup \{aabb\} = \{\varepsilon, ab, aabb\}; \\
&\vdots \\
R^{(i+1)} &= R^{(i)} \cup \{a^n b^n | n = i\}; \\
&\vdots
\end{aligned}$$

Вважаємо, що $R = \cup_{i \in \omega} R^{(i)}$.

(Тут $\omega$ є першим граничним ординаром і може тлумачитись як множина натуральних чисел. Детальніше про це буде сказано в наступному розділі підручника.)

**Лема (про неперервність $\varphi$).** $\varphi(\cup_{i\in\omega}R^{(i)}) = \cup_{i\in\omega}\varphi(R^{(i)})$.

*Доведення.* Спочатку доведемо, що $\varphi(\cup_{i\in\omega}R(i)) \subseteq \cup_{i\in\omega}\varphi(R^{(i)})$. Нехай $x \in \varphi(\cup_{i\in\omega}R^{(i)})$. Тоді $x \in \{\varepsilon\}$ або $x \in \{a\}(\cup_{i\in\omega}R^{(i)})\{b\}$. У першому випадку очевидно, що $x \in \cup_{i\in\omega}\varphi(R^{(i)})$. У другому випадку $x \in \varphi(R^{(i+1)})$, тому $x \in \cup_{i\in\omega}\varphi(R^{(i)})$.

Тепер доведемо, що $\varphi(\cup_{i\in\omega}R^{(i)}) \supseteq \cup_{i\in\omega}\varphi(R^{(i)})$. Нехай $x \in \cup_{i\in\omega}\varphi(R^{(i)})$. Тоді існує $k \in \omega$ таке, що $x \in \varphi(R^{(k)})$. Це означає, що $x \in \{\varepsilon\}$ або $x \in \{a\}(R^{(k-1)})\{b\}$. Тому $x \in \{a\}(\cup_{i\in\omega}R^{(i)})\{b\}$, тобто $x \in \varphi(\cup_{i\in\omega}R^{(i)})$.

Умова неперервності фактично свідчить, що $R = \cup_{i\in\omega}\varphi(R^{(i)})$ є розв'язком рівняння $S = \{\varepsilon\} \cup \{a\}S\{b\}$. Дійсно, для $i \in \omega$ маємо, що $\varphi(R^{(i)}) = R^{(i+1)}$, тому $\bigcup_{i\in\omega}\varphi(R(i)) = \bigcup_{i\in\omega}R(i+1) = \bigcup_{i\in\omega}R(i)$. Остання рівність випливає з того, що $R^{(0)} = \varnothing$, отже, $R = \varphi(R)$.

Неважко довести, що цей розв'язок збігається з мовою, породженою граматикою $G$.

**Лема.** $L(G) = R$.

*Доведення виконуємо індукцією.* Індуктивне твердження таке: нехай $L(G)_k$ – усі термінальні слова, що виводяться із $S$ виведеннями довжиною не більше $k$, тоді $L(G)_k = R(k)$.

*База індукції.* Для $k = 0$ маємо $L(G)_0 = R(0) = \emptyset$.

*Крок індукції.* Нехай індуктивна гіпотеза справедлива для довільного $k$. Доведемо її справедливість для $k+1$.

Умова неперервності фактично свідчить, що $R = \cup_{i\in\omega}\varphi(R^{(i)})$ є розв'язком рівняння $S = \{\varepsilon\} \cup \{a\}S\{b\}$.

**Лема.** $L(G) = R$.

*Доведення.* Виконуємо індукцією по числу кроків виведення у граматиці $G$. База індукції та крок індукції доводять відповідність між мовою породженою граматикою $G$ і мовою $R$. $\square$

У цьому прикладі ми знайшли єдиний розв'язок рівняння. Леми демонструють, що такий розв'язок є єдиним і збігається з мовою, породженою граматикою $G$.

### Практичне завдання

Побудувати КВ граматику $G$ та систему рівнянь $E$ для мови $L = \{c{\cdot}a^n{\cdot}b^{3n} \mid n \geq 0\}$. Довести незалежно, що $L(G) = L$ та $R(E) = L$ ($R(E)$ – розв'язок системи рівнянь $E$).

**Граматика:**

$$P1 : S \to cA$$
$$P2 : A \to aAbbb$$
$$P3 : A \to \varepsilon$$

**Система рівнянь:**

$$S_k = \{c\}A$$
$$A_{k+1} = \{a\}A\{bbb\} \cup \{\varepsilon\}$$

Можна побачити, що будь-який вивід в граматиці має наступну структуру – одне застосування правила P1, потім дека кінцева кількість застосувань правила P2, після чого йде застосування правила P3 і словоформа більше не містить **нетерміналів**.

Таким чином, можна довести наступне твердження:

Нехай $S \Rightarrow^k \alpha$, $k \geq 1$, $\alpha = ca^{k-1}Ab^{3k-3} \vee \alpha = ca^{k-2}b^{3k-6} \wedge k \geq 2$.

Доведемо це за допомогою математичної індукції.

База індукції:

При $k = 1$, можливий лише один вивід – застосування правила P1, отже $\alpha = cA$ твердження виконується.

Крок індукції:

Припустимо, що для $k \leq n$ все вірно. Доведемо для $n + 1$.

Маємо $S \Rightarrow^{n+1} \alpha^{'}$, нехай на попередньому кроці було $S \Rightarrow^n \alpha$, тоді за припущенням $\alpha = ca^{n-1}Ab^{3n-3} \vee \alpha = ca^{n-2}b^{3n-6} \wedge n \geq 2$. Щоб $\alpha \Rightarrow \alpha^{'}$, підходить лише один випадок: $\alpha = ca^{n-1}Ab^{3n-3}$. В цьому разі можливі два випадки – $\alpha \overset{\text{P2}}{\Longrightarrow} \alpha^{'}$ та $\alpha \overset{\text{P3}}{\Longrightarrow} \alpha^{'}$. В першому випадку $\alpha^{`} = ca^nAb^{3n}$, а в другому $\alpha^{`} = ca^{n-1}Ab^{3n-3}$, що й треба було довести. Отже, твердження істинне.

З твердження випливає, що всі слова, що виводяться в граматиці мають вигляд $c \cdot a^k \cdot b^{3k}$, отже належать мові. Разом з твердження випливає, що слово $c \cdot a^k \cdot b^{3k}$ мови може бути виведено за $k + 2$ кроки.

**Виконаємо декілька ітерацій пошуку розв'язку рівняння:**

$S_0 = \varnothing, A_0 = \varnothing$

$S_1 = \{c\}, A_0 = \{c\} * \varnothing = \varnothing$

$A_1 = \{a\}A_0\{bbb\} \cup \{\varepsilon\} = \{a\} * \varnothing * \{bbb\} \cup \{\varepsilon\} = \{\varepsilon\}$

$S_2 = \{c\}, A_1 = \{c\} * \{\varepsilon\} = \{c\}$

$A_2 = \{a\}A_1\{bbb\} \cup \{\varepsilon\} = \{a\} * \{\varepsilon\} * \{bbb\} \cup \{\varepsilon\} = \{\varepsilon, abbb\}$

Стабілізація не відбувається, але можна сформулювати твердження відносно $S_k$, $A_k$.

$S_k = \{ca^ib^{3i} | 0 \leq i < k - 1\}, A_k = \{a^ib^{3i} | 0 \leq i < k\}$ Доведемо це за допомогою математичної індукції.

**База індукції** очевидно виконується.

**Крок індукції:** Припустимо, що для $k \leq n$ твердження вірне, доведемо для $n + 1$.

$S_{n+1} = \{c\}A_n = \{c\} \cdot \{a^ib^{3i} \mid 0 \leq i < n\} = \{ca^ib^{3i} \mid 0 \leq i < n\}$

$A_{n+1} = \{a\}A_n\{\text{bbb}\} \cup \{\varepsilon\} = \{a\} \cdot \{a^ib^{3i} \mid 0 \leq i < n\} \cdot \{\text{bbb}\} \cup \{\varepsilon\} = \{a^{i+1}b^{3i}\text{bbb} \mid 0 \leq i < n\} \cup \{\varepsilon\} = \{a^ib^{3i} \mid 1 \leq i < n + 1\} \cup \{\varepsilon\} = \{a^ib^{3i} \mid 0 \leq i <$

$n+1\}$

Отже, твердження вірне для всіх $k$.

Тоді маємо: $R(E) = \bigcup\limits_k S_k = \{ca^i b^{3i} \mid 0 \le i < k-1\} = \{ca^i b^{3i} \mid 0 \le i\} = L$, що і треба було довести.

**Перелік задач.**

Побудувати КВ граматику $G$ та систему рівнянь $E$ для мови $L$. Довести незалежно, що $L(G) = L$ та $R(E) = L$ ($R(E)$ – розв'язок системи рівнянь $E$).

1. $L = \{e^n b^2 c^{3n} \mid n \ge 0\}$
2. $L = \{e^{2n} d b^{3n} \mid n \ge 0\}$
3. $L = \{a^3 b^n c^{2n} \mid n \ge 0\}$
4. $L = \{a^n b^{3n} d^2 \mid n \ge 0\}$
5. $L = \{e^{3n} d b^n c^2 \mid n \ge 0\}$
6. $L = \{a^{2n} b d c^{3n} \mid n \ge 0\}$
7. $L = \{a e^{3n} d b^{2n} \mid n \ge 0\}$
8. $L = \{b^{2n} d c^{2n} \mid n \ge 0\}$
9. $L = \{a e^{2n} b d^n c^2 \mid n \ge 0\}$
10. $L = \{d^{3n} b^{2n} e \mid n \ge 0\}$
11. $L = \{b^n d c^{2n} \mid n \ge 0\}$
12. $L = \{a^{2n} c d^n \mid n \ge 0\}$
13. $L = \{a d^{3n} b^2 e c^n \mid n \ge 0\}$
14. $L = \{a c b^{2n} c^{3n} \mid n \ge 0\}$
15. $L = \{a^3 d^{2n} c^n \mid n \ge 0\}$
16. $L = \{a b^{3n} d c^{2n} e \mid n \ge 0\}$

## 2.3 ПРАКТИЧНА РОБОТА 6. Побудова граматик за мовою. Лема про накачку

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Синтактика: формальні мови та граматики.
Властивості контекстно-вільних мов. Операції над формальними мовами. Дерева виводу. Однозначні та неоднозначні граматики. (сторінка 90-94).

Лема про накачку (лема про розростання). Нехай $L$ – КВ-мова над алфавітом $T$. Тоді знайдеться таке натуральне число $k$, що для довільного ланцюжка $t \in L$ довжини не менше $k$ знайдуться ланцюжки $u, v, t_1, t_2, x \in T^*$, для яких вірно $ut_1xt_2v = t, |t_1t_2| \geq 1, |t_1xt_2| \leq k$ та $ut_1{}^ixt_2{}^iv \in L$ для всіх $i = 0, 1, \ldots$.

Ідея доведення полягає в тому, що для породження мови $L$ розглядається граматика у нормальній формі Хомського (нескорочуюча граматика). Тоді для достатньо довгих виводів можна виділити рекурсивний нетермінал $A$ такий, що має місце $A \Rightarrow^* t_1t_2$, де $t_1, t_2 \in T*$ та $|t_1t_2| \geq 1$.

Лема. Мова $L3 = \{a^nb^nc^n | n \geq 0\}$ не є КВ-мовою.
Доведення. Якби мова $L3$ була б КВ-мовою, то тоді існували б ланцюжки $u, v, t_1, t_2, x \in \{a, b, c\}^*$ такі, що $ut_1{}^ixt_2{}^iv \in L3$ для всіх $i = 0, 1, \ldots$. Зрозуміло, що $t_1$ (так само як і $t_2$) не може складатися з різних символів (інакше для деякого і ланцюжок $ut_1{}^ixt_2{}^iv$ не буде належати $L3$. Але якщо $t_1$ складається лише з одного символу, то збільшуючи і, можна порушити баланс символів $a, b, c$. Тому мова $L3$ не може бути КВ-мовою.

**Практичне завдання**

Побудувати породжуючу граматику $G$ для мови $L$. Довести, що $L(G) = L$ та що мова $L$ не є КВ мовою. $L = \{db^na^{3n}ec^{3n}b | n \geq 0\}$.

**Побудова граматики** $G$ :
$P_1 : S \to dS'b$
$P_2 : S' \to bAAAS'ccc$
$P_3 : S' \to e$
$P_4 : Ab \to bA$
$P_5 : Ae \to ae$
$P_6 : Aa \to aa$
Правило $P_1$ дозволяє звести проблему до побудови граматики для мови

$L' = \{b^n a^{3n} e c^{3n} | n \geq 0\}$, замінивши початковий нетермінал в якій на $S'$ і отримуємо правила $P_2 - P_6$. Використання правила $P_2$ гарантує, що співвідношення кількості символів $b, A, c$ буде зберігатися. Проте при застосуванні лише даного правила, порядок літер $b, A$ не буде співпадати з тим, що відповідає умові задачі, тому в правилі й використовується нетермінальний символ $A$, що в подальшому буде замінено на термінальний символ $a$. Правило $P_3$ дозволяє вивести символ $e$, що розташовано в словах мови між символами $a$ і $c$. Правило $P_4$ необхідно для впорядкування символів $A$ та $b$. Правила $P_5$ та $P_6$ гарантують, що лише ті нетермінальні символи $A$, що розташовані праворуч від термінальних символів $b$ будуть замінені термінальними $a$, адже символ $e$ є тим контекстом, що вказує на те, що це саме крайній правий символ $A$, праворуч від якого відсутні символи $b$.

Можна привести наступні еквівалентні граматики, що також задають мову $L$. В граматиці $G'$ - нетермінал $A$ замінює послідовність $aaa$ а не єдиний термінал $a$. В граматиці $G''$ правила застосовуються для послідовності з трьох символів $A$. В граматиці $G'''$ перетворення $A$ на $a$ виконується за допомогою єдиного правила.

| $G'$ : | $G''$ : | $G'''$ : |
|---|---|---|
| $S \to dS'b$ | $S \to dS'b$ | $S \to dS'b$ |
| $S' \to bAS'ccc$ | $S' \to bAAAS'ccc$ | $S' \to bS'AAAccc$ |
| $S' \to e$ | $S' \to e$ | $cA \to Ac$ |
| $Ab \to bA$ | $AAAb \to bAAA$ | $S' \to e$ |
| $Ae \to aaae$ | $AAAe \to aaae$ | $eA \to ae$ |
| $Aa \to aaaa$ | $AAAa \to aaaa$ | |

**Доведення** $L \subseteq L(G)$

Покажемо, що для довільного $n \in N$ слово $db^n a^{3n} e c^{3n} b$ може бути виведено в граматиці $G$ (числом під стрілкою виводу будемо позначати скільки разів було застосовано правило):

$S \overset{P_1}{\Longrightarrow} dS'b \underset{n}{\overset{P_2}{\Longrightarrow}} d(bAAA)^n S' c^{3n} b \overset{P_3}{\Longrightarrow} d(bAAA)^n e c^{3n} b \underset{3n(n-1)/2}{\overset{P_4}{\Longrightarrow}} db^n A^{3n} e c^{3n} b \overset{P_5}{\Longrightarrow}$

$db^n A^{3n-1} a e c^{3n} b \underset{3n-1}{\overset{P_6}{\Longrightarrow}} db^n a^{3n} e c^{3n} b$

Отримали необхідне слово, отже довільне слово в мові $L$ може бути виведено в граматиці $G$, тому $L \subseteq L(G)$.

**Доведення** $L(G) \subseteq L$

Доведемо включення, показавши, що будь-яке слово, що виводиться в граматиці $G$ буде належати мові $L$.

Нехай $S \Rightarrow^* \alpha$. Де $\alpha$ деяка словоформа.Доведемо наступні твердження використовуючи ММІ за довжиною виводу.

I $|\alpha|_{b,S} = n+1 \Rightarrow |\alpha|_{A,a} = 3n \wedge |\alpha|_c = 3n$

II $\alpha = S \vee \alpha = d\alpha_1\alpha_2 b,$ де $\alpha_1 \in \{b, A\}^* \wedge (\alpha_2 = S'c^m \vee \alpha_2 = a^l e c^m)$

**База індукції:**
$S \Rightarrow^0 S, \alpha = S$

Твердження I. виконується.
$|S|_c = 0, |S|_{A,a} = 0, |S|_{b,S} = 1$

Твердження II. виконується.
$\alpha = S$

**Крок індукції:** нехай для довільного $S \Rightarrow^k \alpha$ твердження виконуються. Доведемо що для $S \Rightarrow^{k+1} \alpha'$ також обидва твердження будуть виконуватись.
Розглянемо правила, що виконувались на $k+1$ кроці:

Правило $P_1$: $\alpha \overset{P_1}{\Longrightarrow} \alpha'$ :

Згідно припущення, може бути застосовано лише якщо $\alpha = S, \alpha' = dS'b$. Перевіримо твердження:

Твердження I. виконується.
$|\alpha'|_c = |dS'b|_c = 0, |\alpha'|_{A,a} = |dS'b|_{A,a} = 0, |\alpha'|_{b,S} = |dS'b|_{b,S} = 1$

Твердження II. виконується.
$\alpha' = dS'b = d\alpha'_1\alpha'_2 b,$ де $\alpha'_1 = \varepsilon, \alpha'_2 = S'$

Правило $P_2$: $\alpha \overset{P_2}{\Longrightarrow} \alpha'$ :

Згідно припущення, може бути застосовано лише якщо $\alpha = d\alpha_1 S'\alpha_{21} b, \alpha_2 = S'\alpha_{21},\ \alpha' = d\alpha_1 bAAAS'ccc\alpha_{21}b$. Перевіримо твердження:

Твердження I. виконується(співвідношення зберігається).
$|\alpha'|_c = |\alpha|_c + 3, |\alpha'|_{A,a} = |\alpha|_{A,a} + 3, |\alpha'|_{b,S} = |\alpha|_{b,S} + 1$

Твердження II. виконується.
$\alpha' = d\alpha'_1\alpha'_2 b,$ де $\alpha'_1 = \alpha_1 bAAA, \alpha'_2 = S'ccc\alpha_{21}$

Правило $P_3$: $\alpha \overset{P_3}{\Longrightarrow} \alpha'$ :

Згідно припущення, може бути застосовано лише якщо $\alpha = d\alpha_1 S'\alpha_{21} b, \alpha_2 = S'\alpha_{21},\ \alpha' = d\alpha_1 e\alpha_{21} b$. Перевіримо твердження:

Твердження I. виконується(співвідношення зберігається).
$|\alpha'|_c = |\alpha|_c, |\alpha'|_{A,a} = |\alpha|_{A,a}, |\alpha'|_{b,S} = |\alpha|_{b,S}$

Твердження II. виконується.

$\alpha' = d\alpha'_1\alpha'_2 b$, де $\alpha'_1 = \alpha_1, \alpha'_2 = e\alpha_{21}$

Правило $P_4$: $\alpha \overset{P_4}{\Longrightarrow} \alpha'$ :

Згідно припущення, може бути застосовано лише якщо $\alpha = d\alpha_{11}Ab\alpha_{12}\alpha_2 b, \alpha_1 = \alpha_{11}Ab\alpha_{12}$, $\alpha' = d\alpha_{11}bA\alpha_{12}\alpha_2 b$. Перевіримо твердження:

Твердження I. виконується(співвідношення зберігається).
$|\alpha'|_c = |\alpha|_c, |\alpha'|_{A,a} = |\alpha|_{A,a}, |\alpha'|_{b,S} = |\alpha|_{b,S}$

Твердження II. виконується.
$\alpha' = d\alpha'_1\alpha'_2 b$, де $\alpha'_1 = \alpha_{11}bA\alpha_{12}$, $\alpha'_2 = \alpha_2$

Правило $P_5$: $\alpha \overset{P_5}{\Longrightarrow} \alpha'$ :

Згідно припущення, може бути застосовано лише якщо $\alpha = d\alpha_{11}Ae\alpha_{21}b, \alpha_1 = \alpha_{11}A$, $\alpha_2 = e\alpha_{21}$, $\alpha' = d\alpha_{11}ae\alpha_{21}b$. Перевіримо твердження:

Твердження I. виконується(співвідношення зберігається).
$|\alpha'|_c = |\alpha|_c, |\alpha'|_{A,a} = |\alpha|_{A,a}, |\alpha'|_{b,S} = |\alpha|_{b,S}$

Твердження II. виконується.
$\alpha' = d\alpha'_1\alpha'_2 b$, де $\alpha'_1 = \alpha_{11}$, $\alpha'_2 = ae\alpha_{21}$

Правило $P_6$: $\alpha \overset{P_6}{\Longrightarrow} \alpha'$ :

Згідно припущення, може бути застосовано лише якщо $\alpha = d\alpha_{11}Aa\alpha_{21}b, \alpha_1 = \alpha_{11}A$, $\alpha_2 = a\alpha_{21}$, $\alpha' = d\alpha_{11}aa\alpha_{21}b$. Перевіримо твердження:

Твердження I. виконується(співвідношення зберігається).
$|\alpha'|_c = |\alpha|_c, |\alpha'|_{A,a} = |\alpha|_{A,a}, |\alpha'|_{b,S} = |\alpha|_{b,S}$

Твердження II. виконується.
$\alpha' = d\alpha'_1\alpha'_2 b$, де $\alpha'_1 = \alpha_{11}$, $\alpha'_2 = aa\alpha_{21}$

Було розглянуто всі правила, отже крок індукції доведено і твердження виконується.

Розгянемо слова, а не просто словоформи, що будуть виводитись в граматиці $\alpha \in T^*$, твердження будуть мати наступний вигляд:

I $|\alpha|_b = n + 1 \Rightarrow |\alpha|_a = 3n \wedge |\alpha|_c = 3n$

II $\alpha = d\alpha_1\alpha_2 b$, де $\alpha_1 \in \{b\}^* \wedge \alpha_2 = a^l e c^m$

Перепишемо твердження II: $\alpha = db^k a^l e c^m b$. З твердження I маємо : $l = 3k, m = 3k$.

Отже, $\alpha = db^k a^{3k} e c^{3k} b \Rightarrow \alpha \in L$. Таким чином, кожне слово, що виводиться в граматиці буде належати мові $L$, що і доводить $L(G) \subseteq L$.

Доведемо, що $L$ не є КВ мовою. Нехай мова $L$ є КВ мовою, тоді згідно з лемою про накачку існує таке натуральне число $k$, що для довільного ланцюжка $t \in L$ довжини не менше $k$ знайдуться ланцюжки $u, v, t_1, t_2, x$ такі, що $ut_1xt_2v = t, |t_1t_2| \geq 1, |t_1xt_2| \leq k$ та $ut_1^ixt_2^iv \in L$ для всіх $i = 0, 1, ....$ Візьмемо таке $n$, що $|db^na^{3n}ec^{3n}b| \geq k$. Для такого слова, що належить мові і має довжину не менше $k$ згідно леми знайдуться підслова $u, v, t_1, t_2, x$, та принаймні одне зі слів $t_1, t_2$ не є порожнім словом. В такому випадку, це слово не може містити фрагментів, що не повторюються в словах мови - префіксу $d$, суфіксу $e$, або закінчення $b$. Якщо ж воно містить одни з символів, що повторюється, то слова вигляду $ut_1^ixt_2^iv$ не можуть належати граматиці, бо це порушить співвідношення кількості відповідних літер. Декілька різних літер, що повторюються, таке підслово також не може містити. Якщо ж обидва підслова $t_1, t_2$ не є порожніми, кожне з них може мітити лише по одній з літер, що повторюються в мові, але таких літер всього три і в такому випадку співвідношення кількості літер теж не відповідатиме мові. Таким чином, підслова з формулювання леми не можуть бути знайдені, лема не виконується, а отже мова $L$ не є КВ мовою.

**Перелік задач.**

Побудувати породжуючу граматику $G$ для мови $L$. Довести, що $L(G) = L$ та що мова $L$ не є КВ мовою.

1. $L = \{e^nb^{2n}c^{3n} \mid n \geq 0\}$
2. $L = \{e^{2n}db^{3n}c^n \mid n \geq 0\}$
3. $L = \{a^{3n}b^ndc^{2n} \mid n \geq 0\}$
4. $L = \{a^nb^{3n}dc^{2n} \mid n \geq 0\}$
5. $L = \{e^{3n}db^nc^{2n} \mid n \geq 0\}$
6. $L = \{a^{2n}b^ndc^{3n} \mid n \geq 0\}$
7. $L = \{ae^{3n}db^{2n}c^n \mid n \geq 0\}$
8. $L = \{a^nb^{2n}dc^{2n} \mid n \geq 0\}$
9. $L = \{ae^{2n}b^nd^nc^{2n} \mid n \geq 0\}$
10. $L = \{d^{3n}b^{2n}ec^n \mid n \geq 0\}$
11. $L = \{e^nb^ndc^{2n} \mid n \geq 0\}$
12. $L = \{a^{2n}b^{3n}cd^n \mid n \geq 0\}$
13. $L = \{ad^{3n}b^{2n}ec^n \mid n \geq 0\}$

14. $L = \{a^n c b^{2n} c^{3n} \mid n \geq 0\}$
15. $L = \{a^{3n} b d^{2n} c^n \mid n \geq 0\}$
16. $L = \{a b^{3n} d c^{2n} e a^n \mid n \geq 0\}$

## 2.4 МОДУЛЬНА КОНТРОЛЬНА РОБОТА 2. Перелік теоретичних питань та базових задач

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Синтактика: формальні мови та граматики (сторінка 60-95).

**Теоретичні питання до контрольної роботи 2:**

1. Як визначається синтаксичний аспект мов?
2. Як формулюється принцип відокремлення і єдності синтаксичного та семантичного аспектів.
3. Охарактеризуйте основні моделі формальних мов.
4. Якими методами подається синтаксис мов програмування?
5. Що таке БНФ?
6. Які є модифікації БНФ?
7. Що таке формальна мова?
8. Визначте основні поняття теорії формальних мов.
9. Що таке породжуючи граматика?
10. Розкрийте інтенсіональні та екстенсіональні аспекти визначення породжуючих граматик.
11. Дайте визначення виводу в граматиці та мови, породженою граматикою.
12. Яким чином доводиться правильність граматики для породження певної мови?
13. Як визначається ієрархія Хомського?
14. Дайте визначення загально-контекстних граматик.
15. Які алгебраїчні операції задані для формальних мов?
16. Які типи автоматів використовують для подання синтаксису мов?
17. Що таке контекстно-вільні граматики?
18. Які властивості контекстно-вільних граматик?
19. Відносно яких операцій клас контекстно-вільних мов є замкненим?
20. Відносно яких операцій клас контекстно-вільних мов не є замкненим?
Дайте визначення різним нормальним формам контекстно-вільних граматик.
21. Сформулюйте розв'язні проблеми для контекстно-вільних граматик.
22. Сформулюйте нерозв'язні проблеми для контекстно-вільних граматик.
23. Як визначається розв'язок системи рівнянь в алгебрах формальних мов?

**Типове завдання модульної контрольної роботи 1:**

1. Побудувати породжуючу граматику G для мови $L = \{ab^{3n}dc^{2n}ea^n \mid n \geq 0\}$. Довести, що $L(G) = L$ та що мова $L$ не є КВ мовою.

2. Побудувати нормальну форму без $\varepsilon$–правил для КВ-граматики з множиною правил: $\{S \to BbB, S \to cCc, S \to AaA, S \to dDd, A \to aA, B \to bB, C \to cC, D \to dD\}$

3. Побудувати КВ граматику $G$ та систему рівнянь $E$ для мови $L = \{c^n a^2 b^n \mid n \geq 0\}$. Довести незалежно, що $L(G) = L$ та $R(E) = L$ ( $R(E)$ – розв'язок системи рівнянь $E$ ).

# 3 РЕКУРСІЯ ТА НАЙМЕНША НЕРУХОМА ТОЧКА. РЕКУРСИВНІ ПРОГРАМИ

## 3.1 ПРАКТИЧНА РОБОТА 7. Рекурсія в мовах програмування. Робота з рекурсивними функціями

**Основні теоретичні відомості**

Нікітченко М.С. Теорія програмування. Частина 1 — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 122 с.
(https://csc.knu.ua/uk/library/books/nikitchenko-7.pdf)
Теорія рекурсії (теорія найменшої нерухомої точки) (сторінка 100-118).

**Практичне завдання**

Побудувати рекурсивну SIPL-функцію для обчислення $x \div y$ $(x, y > 0)$.

Базовим випадком рекурсії в такому разі будуть значення $x < y$, коли $x \div y = 0$. Рекурсивний перехід визначається тим, що при $x \geq y$ має місце рівність $x \div y = (x - y) \div y + 1$. Тому однією з рекурсивних функцій, що задають частку від ділення $x$ на $y$ є:

func f(X, Y)=if X < Y then 0 else f(X − Y, Y) + 1

Побудуємо семантичний терм для зазначеної рекурсивної програми.

$\phi(f) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(f, S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow)\ ), \overline{1}))$

Побудуємо 3 апроксимації. Найпершою апроксимацією є всюди невизначена функція $f_0 = \perp$.

Наступна апроксимація визначається як $f_1 = \phi(f_0) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(f_0, S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1})) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(\perp, S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1}))$. Зважаючи на те, що суперпозиція одним з аргументів якої є всюди невизначена функція в результаті повертає всюди невизначену функцію, можна виконати наступні спрощення.
$IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(\perp, S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1})) =$
$IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, \perp, \overline{1})) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \perp)$

Третя апроксимація має наступний вигляд: $f_2 = \phi(f_1) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(f_1, S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1})) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \perp), S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1}))$

Аналізуючи отримані апроксимації, можна описати функції наступним чином:

$$f_1(st) = \begin{cases} 0 & \text{якщо } st(X) < st(Y) \\ \bot & \text{інакше} \end{cases}$$

$$f_2(st) = \begin{cases} 0 & \text{якщо } st(X) < st(Y) \\ 1 & \text{якщо } st(Y) \leq st(X) < 2\,st(Y) \\ \bot & \text{інакше} \end{cases}$$

Проведемо тестування побудованих апроксимацій на вхідних даних $x = 7, y = 4$. Для цього застосуємо відповідні семантичні терми до стану $st = [X \mapsto 7, Y \mapsto 4]$.

**Перша апроксимація:**

$f_0(st) = \bot\ (st)\uparrow$

Апроксимація невизначена на вказаних тестових даних, тому потрібно розглянути наступну апроксимацію.

**Друга апроксимація:**

$f_1(st) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot)(st)$

Спочатку обчислимо значення умови:

$S^2(less, X\Rightarrow, Y\Rightarrow)(st) = less(X\Rightarrow (st), Y\Rightarrow (st)) = less(7, 4) = false$

Умова хибна і тому $f_1(st) = \bot\ (st)\uparrow$

Апроксимація невизначена на вказаних тестових даних, тому потрібно розглянути наступну апроксимацію.

**Третя апроксимація:**

$f_2(st) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot), S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1}))(st)$

Умова оператора IF співпадає з умовою в попередній апроксимації, тому маємо:

$f_2(st) = S^2(add, S^{X,Y}(IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot), S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1})(st) = add(S^{X,Y}(IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot), S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow)(st), \overline{1}(st)) = add(S^{X,Y}(IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot), S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow)(st), 1)$

Обчислимо окремо перший аргумент додавання: $S^{X,Y}(IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot), S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow)(st) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot)(st\nabla[X \mapsto S^2(sub, X\Rightarrow, Y\Rightarrow)(st), Y \mapsto Y\Rightarrow (st)]) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot)(st\nabla[X \mapsto sub(7, 4), Y \mapsto 4]) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot)([X \mapsto 3, Y \mapsto 4])$ .

Умова в такому випадку рівна: $S^2(less, X\Rightarrow, Y\Rightarrow)([X \mapsto 3, Y \mapsto 4]) = less(3, 4) = true$, а отже: $IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, \bot)([X \mapsto 3, Y \mapsto 4]) = \overline{0}([X \mapsto 3, Y \mapsto 4]) = 0$

Підставляючи цей резуультат, отримуємо: $f_2(st) = add(0, 1) = 1$. Апроксимація визначена і результат співпадає з $7 \div 4$, отже, тестування завершено успішно.

Доведемо правильність рекурсивної програми.

За наведеними вище апроксимаціями можна припустити, що в загальному вигляді $k$-та апроксимація для $k \geq 1$ буде мати наступний вигляд:

$$f_k(st) = \begin{cases} 0 & \text{якщо } st(X) < st(Y) \\ st(X) \div st(Y) & \text{якщо } st(Y) \leq st(X) < k\, st(Y) \\ \bot & \text{інакше} \end{cases}$$

Доведемо це твердження за допомогою математичної індукції.

**База індукції:**

Перевіримо базу індукції для $k = 1$. Порівняємо побудовані раніше апроксимації з припущенням.

$$f_1(st) = \begin{cases} 0 & \text{якщо } st(X) < st(Y) \\ st(X) \div st(Y) & \text{якщо } st(Y) \leq st(X) < st(Y) \\ \bot & \text{інакше} \end{cases}$$

Враховуючи, що $st(X) \div st(Y) = 0$ при додатніх $st(X) < st(Y)$, співпадає з раніше побудованою $f_1$.

**Крок індукції:**

Припустимо, що твердження виконується для $k \leq n, 1 \leq n$, доведемо, що воно виконується для $k = n + 1$. Потрібно показати, що:

$$f_{n+1}(st) = \begin{cases} 0 & \text{якщо } st(X) < st(Y) \\ st(X) \div st(Y) & \text{якщо } st(Y) \leq st(X) < (n+1)\, st(Y) \\ \bot & \text{інакше} \end{cases}$$

Враховуючи неперервність та монотонність оператора $\phi$ маємо, що якщо $f_n(st)\downarrow$, то $f_{n+1}(st)\downarrow = f_n(st)$. Це доводить потрібне твердження для випадку $st(X) < n\, st(Y)$. Залишається лише розглянути випадки $n\, st(Y) \leq st(X) < (n+1)\, st(Y)$ та $(n+1)\, st(Y) \leq st(X)$.

Нехай $st = [X \mapsto x, Y \mapsto y]$, обчислимо апроксимацію $f_{n+1}$ на такому стані:

$f_{n+1}(st) = \phi(f_n)(st) = IF(S^2(less, X\Rightarrow, Y\Rightarrow), \overline{0}, S^2(add, S^{X,Y}(f_n, S^2(sub, X\Rightarrow$ $, Y\Rightarrow), Y\Rightarrow), \overline{1}))(st)$

Обчислимо умову: $S^2(less, X\Rightarrow, Y\Rightarrow)(st) = less(x, y)$. У випадках, що нас цікавлять ($n\, y \leq x < (n+1)\, y$ та $(n+1)\, y \leq x$) дана умова буде хибною. Тому $f_{n+1}(st) = S^2(add, S^{X,Y}(f_n, S^2(sub, X\Rightarrow, Y\Rightarrow), Y\Rightarrow), \overline{1})(st) = add(S^{X,Y}(f_n, S^2(sub, X\Rightarrow$ $, Y\Rightarrow), Y\Rightarrow)(st), 1) = add(f_n(st\nabla[X \mapsto sub(X\Rightarrow (st), Y\Rightarrow (st), Y \mapsto Y\Rightarrow$ $(st)], 1) = add(f_n(st\nabla[X \mapsto x - y, Y \mapsto y], 1)$.

Розглянемо випадок $n\, y \leq x < (n+1)\, y$, в такому разі $x - y < n\, y$, тому апроксимація на відповідних даних визначена і $add(f_n(st\nabla[X \mapsto x - y, Y \mapsto$

$y], 1) = add((x - y) \div y, 1) = (x - y) \div y + 1 = x \div y$. Що і потрібно було показати для цього випадку.

Розглянемо випадок $(n + 1)\, y \leq x$, тепер $n\, y \leq x - y$, тому апроксимація $f_n$ на відповідних даних невизначена. Тому невизначеним є і додавання і результат $f_{n+1}$. Це відповідає твердженню, що потрібно показати.

Було розглянуто всі варіанти вхідних даних, отже твердження доведено. Розглядаючи ланцюг апроксимації, функція, що задається рекурсивним визначенням співпадає з границею ланцюга

$$f(st) = \bigsqcup_k f_k(st) = \begin{cases} 0 & \text{якщо } st(X) < st(Y) \\ st(X) \div st(Y) & \text{якщо } st(Y) \leq st(X) \end{cases}$$

Дана функція співпадає з $x \div y$ для $(x, y > 0)$. Таким чином, правильність рекурсивної програми доведено.

**Проєктна робота.**

1. Розширити БНФ мови SIPL для роботи з рекурсивними функціями.
2. Розв'язати задачу (обравши задачу з переліку).
3. Навести перелік та коротко розкрити теоретичні питання, що використовуються при розв'язанні задачі.
4. Робота має бути оформлена в текстовому редакторі (за вибором), захищена на парі та здана в Google-класі.

## Перелік задач

**Завдання:**

1. Написати рекурсивну SIPL-функцію.
2. Побудувати семантичний терм.
3. Побудувати 3 апроксимації.
4. Провести тестування побудованих апроксимацій на вхідних даних.
5. Довести правильність рекурсивної програми.

**Задачі:**

1. Написати програму обчислення $x - y$, використовуючи функцію $-$ $(x, y > 0)$. Вхідні дані: $x = 4, y = 1$.
2. Написати програму обчислення $x \mod y$ $(x, y > 0)$. Вхідні дані: $x = 3, y = 2$.

3. Написати програму обчислення суми арифметичної прогресії $2, 4, \ldots, 2n$, використовуючи функції $+$, $-$ $(n > 0)$. Вхідні дані: $n = 1$.

4. Написати програму обчислення $x^y$, використовуючи функції $*$, $+$, $-$ $(x, y > 0)$. Вхідні дані: $x = 5, y = 2$.

5. Написати програму обчислення $[\log_x y]$, використовуючи функції `div`, `mod`, $+$, $-$ $(x > 1, y > 0)$. Вхідні дані: $x = 3, y = 7$.

6. Написати програму перевірки чи ділиться $x$ на $y$, використовуючи функції $+$, $-$ $(x, y > 0)$. Вхідні дані: $x = 9, y = 5$.

7. Написати програму обчислення $3^x$, використовуючи функції $*$, $+$, - $(x > 0)$. Вхідні дані: $x = 2$.

8. Написати програму обчислення $[\log_2 n]$, використовуючи функції `div`, `mod`, $+$, $-$, $(n > 0)$. Вхідні дані: $n = 3$.

9. Написати програму обчислення $(2n)!!$ $(n > 0)$. Вхідні дані: $n = 2$.

10. Написати програму обчислення $(2n + 1)!!$ $(n > 0)$. Вхідні дані: $n = 2$.

11. Для обчислення суми геометричної прогресії $1, 2, 4, \ldots, 2^n$, використовуючи функції $*, +, -$ $(n > 0)$. Вхідні дані: $n = 3$.

12. Для перевірки парності числа $n$, за допомогою циклу, не використовуючи функції `div`, `mod` $(n > 0)$. Вхідні дані: $n = 5$.

13. Для обчислення суми арифметичної прогресії $3, 6, \ldots, 3n$, використовуючи функції $+$, $-$ $(n > 0)$. Вхідні дані: $n = 3$.

14. Для обчислення суми геометричної прогресії $1, 3, 9, \ldots, 3^n$, використовуючи функції $*, +, -$ $(n > 0)$. Вхідні дані: $n = 3$.

**Контрольні запитання**

1. Що таке рекурсивне визначення (рекурсивне рівняння)?

2. Які парадокси пов'язані з рекурсивними визначеннями?

3. Який метод застосовують для розв'язання рекурсивних рівнянь? Які поняття пов'язані з цим методом?

4. Що таке перший граничний (перший нескінченний) ординал?

5. Що таке $\omega$-область?

6. Наведіть визначення неперервного відображення.

7. Сформулюйте теорему Кнастера–Тарського–Кліні.

8. Яку структуру має множина нерухомих точок неперервного оператора?

9. Визначте методи конструювання похідних областей.
10. Сформулюйте властивості оператора найменшої нерухомої точки.
11. Як теорія ННТ застосовується для подання синтаксису мов програмування?
12. Як теорія ННТ застосовується для подання семантики мов програмування?
13. Як визначаються рекурсивні розширення мови SIPL?
14. Наведіть приклади рекурсивних визначень у розширеннях мови SIPL.

## РЕКОМЕНДОВАНІ ДЖЕРЕЛА

### Основна

1. М.С. Нікітченко. Теорія програмування. Частина 1. Навчальний посібник. – Ніжин. Видавництво НДУ імені М.В. Гоголя, 2010.– 122 с.

### Додаткова

1. Eric C.R. Hehner. a Practical Theory of Programming. - Springer-Verlag Publishers, New York, 2021. – 243 p.

2. M. Balaban. Principles of Programming Languages. - Ben-Gurion University of the Negev Faculty of Natural Science Department of Computer Science, 2017. – 418 p.

3. Semantics - Advances in Theories and Mathematical Models. - IN-TECH, 2012. – 284 p.

4. J. V. Tucker, K. Stephenson. Data, Syntax and Semantics: An Introduction to Modelling Programming Languages. - University of Wales Swansea, 2006. – 840 p.